**Submitted to *Operations Research***
**manuscript (Please, provide the manuscript number!)**



# The Block Retrieval Problem

Marcos de Melo da Silva
LIPN (UMR CNRS 7030) - Institut Galilée - Université Paris-Nord, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France, marcos.demelodasilva@lipn.univ-paris13.fr

Güneş Erdoğan
School of Management - University of Bath, Bath, BA2 7AY, United Kingdom, g.erdogan@bath.ac.uk

Maria Battarra
School of Management - University of Bath, Bath, BA2 7AY, United Kingdom, m.battarra@bath.ac.uk

Vitaly Strusevich
Department of Mathematical Sciences - University of Greenwich, Park Row, London SE10 9LS, United Kingdom, V.Strusevich@greenwich.ac.uk

Retrieving containers from a bay in a port terminal yard is a time consuming activity. The *Block Retrieval Problem* (BRTP) aims to minimize the number of *relocations*, the unproductive moves of hindering containers, while retrieving target containers belonging to a customer. The choice of relocations leads to alternative bay configurations, some of which would minimize the relocations of forthcoming retrievals. The Bi-objective Block Retrieval Problem (2BRTP) includes a secondary objective, the minimization of the expected number of relocations for retrieving the containers of the next customer. This paper provides $\mathcal{NP}$-Hardness proofs for both the BRTP and 2BRTP. A branch-and-bound (B&B) algorithm and a linear time heuristic are developed for the BRTP; a B&B algorithm and a beam search algorithm are presented for the 2BRTP. Extensive computational tests on randomly generated instances as well as instances adapted from the literature are performed, and the results are presented.

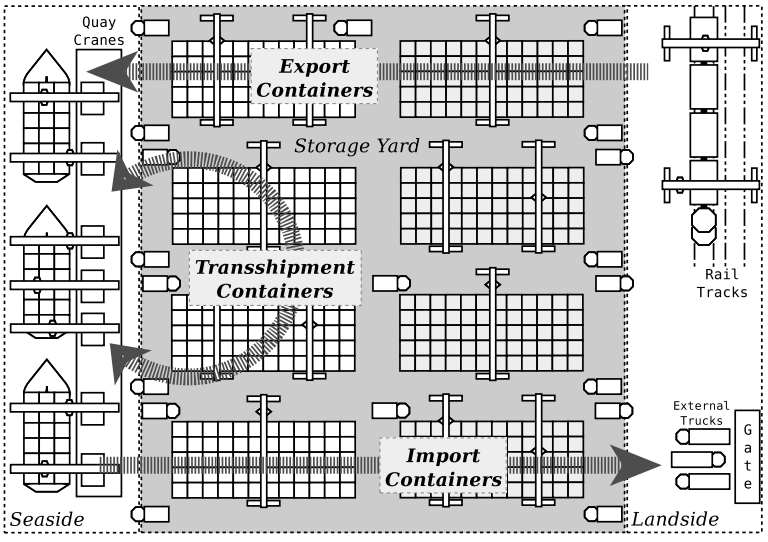
*Key words*: container terminals, import containers, Branch & Bound

*History*: This paper was first submitted on August 18$^{st}$, 2016.

## 1. Introduction

Container terminals are exchange hubs for containers flowing from one transportation mode to another, or between container ships. Container terminals are typically located at ports, where containers are loaded to/unloaded from cargo boats and delivered to the customers for the last mile transportation. Containers can be classified as *export* (or *outbound*) containers, *import* or (*inbound*) containers, and *transshipment* containers (Kim and Park 2003, Caserta et al. 2011). Export containers arrive by trucks or trains to the terminal landside area, then they are placed in the storage yard by internal vehicles, and relocated to the seaside area when the corresponding cargo boat is available. Import containers follow the reverse path; they arrive in the port by cargo boats, then they are unloaded by quay cranes, and placed in the storage yard to be picked up by trucks or trains in the landside area. Transshipment containers are restricted to the seaside and the storage yard areas; they are unloaded from a vessel and stored in the port yard until they are loaded to another cargo boat. A schematic representation of a container terminal together with a classification of container flows is depicted in Figure 1.

**Figure 1** A typical container terminal and the three types of container flows.

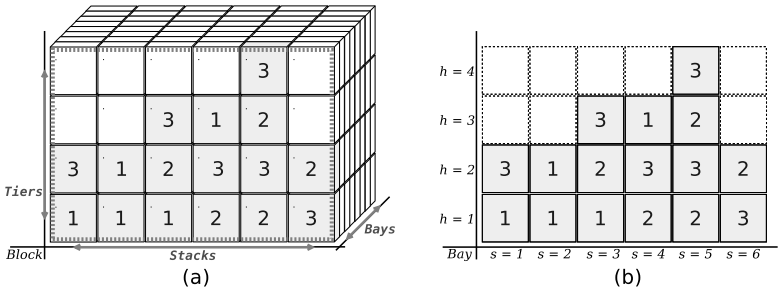



Container terminals provide temporary storage space for preventing the need of synchronization between the transportation modes, in addition to their function as exchange areas. Thus, containers

arriving in the terminal by ship, or by external trucks, or by trains are temporarily kept in storage areas until they are requested for shipment. The dwell times for export, import, and transshipment containers are different. Export containers typically arrive in the port up to weeks before the time they have to be shipped and import containers stay in the terminal yard until they are claimed by the customers.

The storage yard is a scarce resource just as berths, cranes, and internal vehicles, and its usage needs to be carefully planned. Containers are stored in stacks in order to avoid spreading them around the terminal yard, which would have required larger areas and would have demanded a more substantial transportation effort. Stacks are aligned to form bays and blocks, as illustrated in Figure 2. This configuration optimizes the space utilization and allows for crane operations. Nevertheless, by adopting this storage policy, a trade-off between space saving and handling effort for loading and unloading operations arises. More precisely, hindering or obstructing containers may need to be relocated in order to provide access to the blocked target containers during a retrieval request.

**Figure 2** (a) Container storage block, and corresponding (b) bay representation.




We assume each bay stores uniformly shaped containers, consisting of $S$ stacks of maximum height $H$. Each container belongs to a group $g \in \{1, \ldots, G\}$ which can be defined according to its weight, destination, owner, or any other classification criterion that can be used during the retrieval process. Containers in a group have to be retrieved at the same time. The *target group* refers to containers of the group $t \in \{1, \ldots, G\}$ which are about to be retrieved. The LIFO policy has to be observed; i.e., only the topmost containers can be directly reached by a crane or a granty.

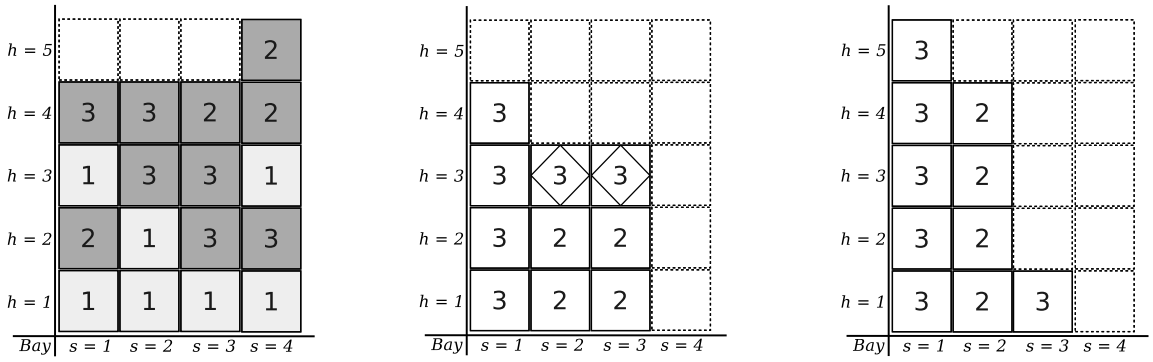
Thus, retrieving a container that is not positioned in the top tier requires all containers above it to be moved to other stacks. Obstructing containers on top of target containers are called *deadlocks*. A *relocation* occurs when a deadlock has to be moved from one stack to another. A *retrieval* is performed when a container of the target group is picked and leaves the bay. It is assumed that containers can be relocated to other stacks and hence there exists enough space above the stacks to perform the necessary relocations. In practice, it is not always possible to assure enough space above the stacks to perform the necessary relocations, in such cases the non-target containers are relocated to a temporary *dummy stack* at the side of the bay. Once the retrieval process is finished, the containers in the dummy stack have to be put back into the bay.

In the *Block Retrieval Problem* (BRTP), the aim is to retrieve all containers of a given target group $t$, $1 \leq t \leq G$, provided that the relocation cost $C_t$ is minimized. In this paper, the cost $C_t$ is defined as the number of relocations needed to retrieve all the containers of the target group $t$. Even if there are several groups other than the target group $t$, in the BRTP there is no distinction between them, i.e., essentially there are only two groups: the given target group $t$ and the other group formed by all remaining containers.

The BRTP is important in its own right; however, its objective is myopic, since it does not consider a configuration of the deadlocks once they have been relocated in the bay, although that could be important for retrievals of the forthcoming target groups. The *Bi-objective Block Retrieval Problem* (2BRTP) overcomes this limitation. The 2BRTP is a bi-criteria problem with two lexicographically ordered objectives: the primary objective is to minimize the cost $C_t$ of the retrieval of the initial target group $t$, and the secondary objective is to minimize the expected number of relocations of the forthcoming retrieval. Assuming that the probability that a group $g$, $1 \leq g \leq G$, $g \neq t$, is retrieved after the first target group $t$ is known and is equal to $P_g$, the secondary objective function can be written as $\sum_{g \in \{1,\ldots,G\} \setminus t} P_g C_g$, where $C_g$ is the cost of the retrieval of group $g$ from the bay of the configuration obtained after the containers of the primary target group $t$ have been retrieved at the minimum cost $C_t$. Figure 3a illustrates an instance of the 2BRTP. In

both solutions shown in Figures 3b and 3c the smallest relocation cost is $C_1 = 12$, i.e., each of these solutions can be considered as an optimal solution to the BRTP. However, for the configuration shown in Figure 3b the secondary objective used for the 2BRTP is $2 \times P_2$, while in the configuration in Figure 3c there are no deadlocks, so that it corresponds to an optimal solution of the 2BRTP.

**Figure 3** (a) A bay with 4 stacks of maximum height 5 holding containers partitioned into 3 groups, target group 1 and obstructing containers in dark gray, (b) BRTP solution where the future deadlocks are diamond shape, (c) 2BRTP optimal solution.




(a) (b) (c)

The remainder of this paper is organized as follows. Section 2 provides a literature review. We present a proof of $\mathcal{NP}$-Hardness for the BRTP (Section 3.1) and we show that the BRTP can be solved in polynomial time if $S$ is constant (Section 3.2). A B&B algorithm for the BRTP (Section 3.3) and a linear algorithm for solving a special class of BRTP instances (Section 3.4) are provided. We present a proof of $\mathcal{NP}$-Hardness for the 2BRTP (Section 4.1) as well as a B&B algorithm (Section 4.2), and a beam search heuristic (Section 4.3). Extensive computational experiments have been performed on newly generated instances and adaptations of instances from the literature, the results of which are in Section 5. Finally, Section 6 provides our concluding remarks.

## 2. Literature review

The turnaround time of vessels, trains and external trucks are key factors to measure the efficiency of container terminals, and also contribute to the evaluation of customer service levels and port

competitiveness (Kim and Kim 1999). Among many factors that may affect the overall turnaround time, loading and unloading operations are the most time consuming tasks performed in terminal yards, and therefore they need to be optimized (Roberti and Pacino 2016). The BRTP and 2BRTP aim at improving the retrieval times, in particular in the landside area of the port terminal. The amount of retrieval information differs for import and export containers (Caserta et al. 2011). The loading sequence on a cargo boat for export containers is typically known when a cargo boat approaches the berth, whereas the retrieval sequence for import containers is typically revealed during the delivery process. Trucks or trains may arrive at the container terminal for the retrieval at unspecified times. The landside retrieval process has been less investigated in the literature. In the following, we provide a summary of related problems.

The BRTP differs from the Blocks Relocation Problem (BRP), another stacking problem arising in storage yards, in which the objective is the same but all groups have to be retrieved in a known sequence (Kim and Hong 2006, Caserta and Voß 2009). Mathematical formulations and complexity results for the BRP are presented in Caserta et al. (2012) and Zehendner et al. (2015). The 2BRTP assumes that the first retrieval is known, but the forthcoming group to be retrieved is uncertain. In terms of container flow in seaports, the BRP better models the retrieval of export containers to be loaded into vessels (in which the loading sequence is deterministic), while the BRTP and 2BRTP suit the retrieval operations of import containers to be delivered to the landside transportation operator (in which the customers' arrival times are not known or are uncertain). A recent survey and classification scheme for loading, unloading, and pre-marshalling operations in stack storage contexts is proposed by Lehnfeld and Knust (2014). For additional contributions and surveys on container terminals and their operations see Steenken et al. (2004), Dekker et al. (2006), Vis and Roodbergen (2009), Carlo et al. (2013, 2014a,b), Gharehgozli et al. (2015).

The number of relocations needed to retrieve target containers from stacks is affected by the height and the width of the stacks and the strategy employed during the storage process. Castilho and Daganzo (1993) analyze two strategies which involve segregation and non-segregation of new

containers arriving to the storage area (i.e., new containers are put in a dedicated area or they are mixed with the previously stored containers). They present methods for estimating the expected number of relocations when retrieving a single container and also when retrieving several containers from a group of stacks in two scenarios, with and without new containers being added to the stacks. Using simulation, the authors observe that in the scenario in which new containers enter the bay, the segregation strategy is advantageous if the throughput of incoming ships is small and the stacks are high. In the case of large ships throughput and short stacks, the non-segregating strategy results in a better performance.

Considering a given bay setting in which no incoming containers are allowed, Kim (1997) proposes a numerical approximation to estimate the expected number of relocations to retrieve an arbitrary container and also to retrieve all containers from the bay. The estimation approach outperforms other methods available (Watanabe 1991) in both accuracy and unbiasedness of the estimations.

The problem addressed in this paper differs from those described in the papers reviewed above. The contributions in the literature focus on the rehandling of import containers at the strategic level; i.e., searching for the best stacking dimensions and storage strategies for reducing or avoiding relocations during the retrieving process. This paper deals with the problem at an operational level, and provides fast algorithms for the retrieval of target containers.

Similarly, Borjian et al. (2015a) and Borjian et al. (2015b) study the Container (Blocks) Relocation Problem and the Blocks Relocation Problem with Incomplete Information. For the latter, the authors assume that the retrieval sequence of some container groups is known, while for the remaining groups only a probability distribution of the retrieval order is given. In addition, the true retrieval sequence of the unknown groups is revealed during the execution of the algorithm. For the BRP, they introduce service time windows among other side constraints, and provide an integer programming formulation. For the BRP with Incomplete Information, the authors adopt an $A^*$ algorithm from the literature and combine it with a sampling technique.

## 3. The Block Retrieval Problem

In this section, we prove that the BRTP is $\mathcal{NP}$-Hard if the number of stacks is variable and admits a polynomial-time solution algorithm for a fixed number of stacks. We also describe a branch-and bound (B&B) algorithm and a linear time algorithm for a restricted set of BRTP instances.

An instance of the BRTP can be described by providing the following information. There are $S \geq 2$ stacks such that stack $i$ contains $h_i$ tiers numbered from bottom to top, i.e., may contain up to $h_i$ containers. Let us call $h_i$ the *height* of stack $i$ and define $H = \max\{h_i | 1 \leq i \leq S\}$, the maximum height. We distinguish between the target containers and non-target containers. Considering a stack $i$ in the order of numbering of its tiers, let $u_{i,0}$ be the number of non-target containers at the bottom of the stack; if there are no such containers then $u_{i,0} = 0$. The remaining containers are organized in $p_i$ pairs of the form $(w_{i,j}, u_{i,j})$, where for the $j$-th pair, $1 \leq j \leq p_i$, $w_{i,j}$ is the number of contiguous slots with the target containers while $u_{i,j}$ is the number of contiguous slots with the non-target containers. Thus, a configuration of stack $i$ can be described by a string of the form

$$Q_i = [u_{i,0}, (w_{i,1}, u_{i,1}), \ldots, (w_{i,p_i}, u_{p_i})].$$

The objective is to retrieve all target containers making the smallest possible number of relocations of non-target containers.

### 3.1. BRTP with Variable Number of Stacks

We prove the $\mathcal{NP}$-Hardness of the BRTP by providing a valid reduction from the well-known $\mathcal{NP}$-Complete problem Partition.

Recall the definition of Partition from Garey and Johnson (1979). Given positive integers $a_1, \ldots, a_n$ and the index set $N = \{1, \ldots, n\}$ such that $\sum_{i \in N} a_i = 2A$, is it possible to partition set $N$ into disjoint subsets $N_1$ and $N_2$ such that $\sum_{i \in N_1} a_i = \sum_{i \in N_2} a_i = A$?

Proposition 1. *The Block Retrieval Problem (BRTP) is $\mathcal{NP}$-Hard in the ordinary sense, provided that the number of stacks $S$ is variable.*

*Proof.* Given an instance of PARTITION, denote

$$\hat{a} = \max\{a_i | i \in N\},$$

$$X = 2n\hat{a} - 3A.$$

Define the following instance of the BRTP. The number of stacks is $S = n+1$ and the height of each of them is $H = 2\hat{a} + X + 1$. Each stack $i$, $1 \leq i \leq n$, is described by the configuration

$$Q_i = [0, (1, X), (a_i, a_i)].$$

Thus, starting from the bottom of stack $i$, its content is a single target container, a set of $X$ non-target containers, $a_i$ target containers and $a_i$ non-target containers. The configuration of stack $n+1$ is given by

$$Q_{n+1} = [0, (1, X)].$$

We show that for the created instance the cost of retrieving all target containers is at most $(n+2)X + 3A$ if and only if PARTITION has a solution. Note that the total number of the deadlocks in this instance is $(n+1)X + 2A$. For this instance, the length of input of PARTITION is bounded above by $n \log \hat{a}$. To compute $A$ and $X$, we need $\mathcal{O}(n)$ operations, and the content of each stack requires at most $\log X + 2\log \hat{a}$ bits, so that the described reduction is polynomial with respect to the length of input of PARTITION.

For an instance of PARTITION with $n = 4$ and $a_1 = 1$, $a_2 = a_3 = 2$ and $a_4 = 3$, the structure of the corresponding instance of the BRTP is shown in Figure 4a. The target containers are labelled by "1" while the non-target containers are labelled by "0"Ẇe will use this example to illustrate parts of the proof.

Necessity ($\Rightarrow$): Assume that PARTITION has a solution, so that $N_1$ and $N_2$ are the required subsets such that $N_1 = \{i_1, \ldots, i_q\}$. Notice that the total number of free slots in all stacks $i$, $1 \leq i \leq n+1$, is equal to $(n+1)(H - X - 1) - 4A = 2(n+1)\hat{a} - 4A = X - A + 2\hat{a}$, including $2\hat{a}$ free slots in stack $n+1$. Perform the following actions:

*1.* For stack $i_1$, move $a_{i_1}$ non-target containers to stack $n+1$ and retrieve $a_{i_1}$ target containers.

*2.* For each $k$ from 2 to $q$, for stack $i_k$, move $a_{i_k}$ non-target containers to stack $i_{k-1}$ and retrieve $a_{i_k}$ target containers.

This transformation requires $\sum_{k=1}^{q} a_{i_k} = A$ moves and creates $A$ additional free slots. As a result of this transformation, there exists at least one stack $i_q$ containing one target container at the bottom and $X$ non-target on top, i.e., its current configuration is $Q_{i_q} = [0, (1, X)]$, and the total number of free slots in all other stacks is equal to $X$. This bay configuration is illustrated in Figure 4b, where we have taken $N_1 = \{1, 4\}$, i.e., $a_1 = 1$ container is moved from stack 1 to stack 5, followed by the relocation of $a_4 = 3$ containers from stack 4 to stack 1.

After actions 1 and 2 are completed, we may proceed as follows.

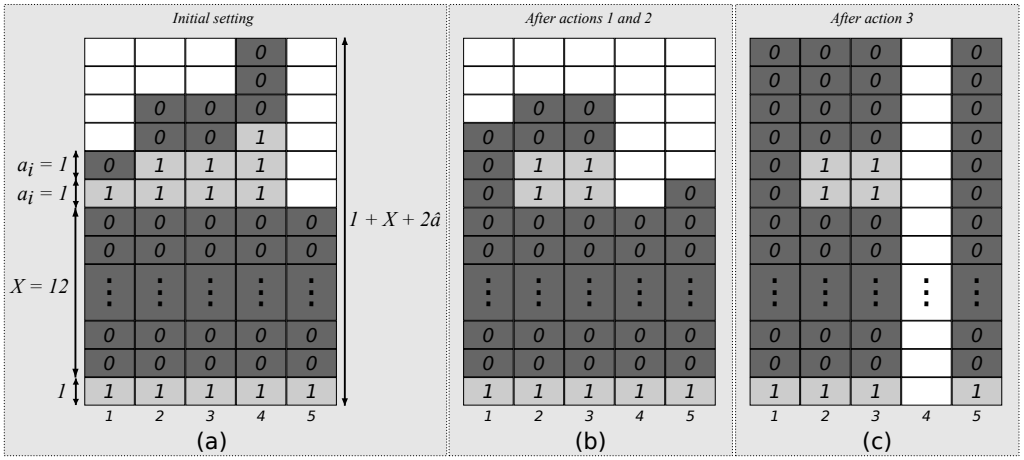
*3.* Move $X$ non-target containers from stack $i_q$ to stacks with available space, make stack $i_q$ empty by retrieving the bottom target container (see Figure 4c).

*4.* Process non-empty stacks with target containers in any order. Empty each stack by moving the non-target containers into an empty stack and retrieving all target containers.

After Step 3, stack $i_q$ is empty, with $H$ free slots, while all other $n$ stacks are completely full. In each iteration of Step 4, a stack with target containers is made empty, and its non-target containers are moved to the stack that is empty in the beginning of the iteration. In Step 4, each of the $(n+1)X + 2A$ non-target containers is relocated exactly once. In the end of this process, all target containers will be retrieved. Together with $A + X$ moves performed in Steps 1, 2 and 3, the total number of moves is $(n+2)X + 3A$.

Sufficiency ($\Leftarrow$): Suppose that for the constructed instance of the BRTP, there exists a solution with at most $(n+2)X + 3A$ moves. We now demonstrate that no solution can be obtained with less than $(n+2)X + 3A$ relocations. During the relocation process, a situation arises that we call Event $R$: some stack $k$ contains only one target container at the bottom and $X$ non-target containers on top, and there is enough room in all other stacks to relocate the $X$ non-target containers from stack $k$. Denote the number of moves that lead to Event $R$ by $T(R)$. If Event $R$ occurs, all target

**Figure 4** BRTP instance for PARTITION with $n = 4$, $a_1 = 1$, $a_2 = a_3 = 2$ and $a_4 = 3$, and bay layouts after performing actions 1, 2, and 3.




containers can be retrieved, starting from stack $k$, in $(n+2)X + 2A$ moves, relocating each non-target container exactly once. If $T(R) > A$, then the total number of moves exceeds $(n+2)X + 3A$, which contradicts our supposition. Thus, $T(R) \leq A$.

Consider the process of reaching Event $R$. We cannot start with the removal of the $X$ non-target containers from stack $n+1$, since the number of the free slots in all other stacks is $X - A$. Thus, we will move non-target containers from the top pair of some stacks $i$, $1 \leq i \leq n$. If Event $R$ is reached after $T(R)$ such moves are performed, then $T(R)$ target containers are retrieved, the total number of all free slots is $X - A + T(R) + 2\hat{a}$, and there exists a stack $k$ with only one target container at the bottom, $X$ non-target on top and $2\hat{a}$ free slots in that stack. In the case of Event $R$, we must have at least $X$ free slots in all stacks other than $k$, which is only possible if $T(R) = A$.

Denote by $N_1$ the set of stacks from which the target containers are retrieved in order to reach Event $R$, and define $N_2 = N \backslash N_1$. These two sets form a solution to PARTITION. $\square$

### 3.2. Fixed Number of Stacks

In this subsection, we show that the BRTP is solvable in polynomial time with respect to the number of containers (or, equivalently, the maximum stack height $H$), provided that the number of stacks is fixed. We will employ a dynamic programming (DP) algorithm for this purpose.

Given an instance of the BRTP represented by $S$ strings $Q_i$, $1 \leq i \leq S$, we redefine $h_i := h_i - u_{i,0}$, since the non-target containers below all target containers are not relocated and only affect the number of free slots in a stack.

As a preprocessing part of the algorithm, for each stack $i$, $1 \leq i \leq S$, compute $f_{i,k}$, the number of slots above the $k$-th pair

$$f_{i,0} = h_i,$$
$$f_{i,k} = f_{i,k-1} - (w_{i,k} + u_{i,k}), \ k = 1, \ldots, p_i.$$

We show that the problem with a fixed number of stacks can be solved by a DP algorithm. The algorithm uses states of the form:

$$(k_1, x_1; k_2, x_2; \ldots; k_S, x_S),$$

where

$k_i$ is the number of pairs in stack $i$ in the current partial solution; if $k_i = 0$ then stack $i$ may only contain non-target containers;

$x_i$ is the number of non-target containers in the top pair in stack $i$; if $k_i = 0$ then $x_i$ is the number of all (non-target) containers in the stack;

For a given state, we denote by $\Phi$ the current number of relocations, i.e., the smallest number of moves of non-target containers needed to achieve that state. The DP algorithm starts with the initial state $(p_1, u_{1,p_1}; p_2, u_{2,p_2}; \cdots; p_S, u_{S,p_S})$ associated with $\Phi(p_1, u_{1,p_1}; p_2, u_{2,p_2}; \cdots; p_S, u_{S,p_S}) = 0$.

Given a state of the form $(k_1, x_1; k_2, x_2; \ldots; k_S, x_S)$, the algorithm selects a stack $t$ from which $x_t$ non-target containers will be relocated. For the case of general $S$, we have to generate all options of redistributing the non-target containers from a chosen stack.

To facilitate these options, recall that a *composition* of an integer $u$ made of $v$ summands is a sequence $(z_1, z_2, \ldots, z_v)$ of positive integers such that $u = z_1 + z_2 + \cdots + z_v$. According to Flajolet and Sedgewick (2009), the number of compositions of $u$ into at most $v$ positive summands (i.e., exactly $v$ non-negative summands) is

$$C_u^{(\leq v)} = \binom{u+v-1}{v-1}, \tag{1}$$

which can be estimated as $\mathcal{O}(u^{v-1})$.

A typical recursion can be written as follows. For a state of the form $(k_1, x_1; k_2, x_2; \ldots; k_S, x_S)$, select a stack $t$ with $k_t \geq 1$. For a generated composition $x_t = z_1 + \cdots + z_{t-1} + z_{t+1} + \cdots + z_S$ into $S-1$ non-negative summands, define the state

$$\left(k_1, x_1 + z_1; \ldots; k_{t-1}, x_{t-1} + z_{t-1}; k_t - 1, u_{t,k_{t-1}}; k_{t+1}, x_{t+1} + z_{t+1}; \ldots; k_S, x_S + z_S\right),$$

which is only feasible if the inequality

$$f_{i,k_i-1} + x_i + z_i \leq h_i$$

holds for each $i$, $1 \leq i \leq S, i \neq t$. Notice that for stack $t$ the number of pairs decreases by 1 since $u_{t,k_t}$ non-target containers are relocated and then $w_{t,k_t}$ target containers are retrieved.

The process is repeated until all states of the form $(0, x_1; 0, x_2; \cdots; 0, x_S)$ are found. If during the described computation, a state with the same values of state variables has already been reached, we keep only one state associated with the smallest value of the function $\Phi$. The smallest value of the function $\Phi$ associated with the states of the form $(0, x_1; 0, x_2; \cdots; 0, x_S)$ is the optimal number of relocations. The sequence of relocations that solves the problem can be found by backtracking.

For a chosen stack $t$, $1 \leq t \leq S$, the overall number of ways to redistribute $x_t$ non-target containers from the its top pair does not exceed the number of compositions of $x_t$ into exactly $S-1$ non-negative summands. Thus, the total number of states generated this way from a given state $(k_1, x_1; k_2, x_2; \ldots; k_S, x_S)$ for a chosen stack $t$ is $\mathcal{O}\left(x_t^{S-2}\right)$, which for all stacks yields $\mathcal{O}(H^{S-2})$. The total number of generated states is $\mathcal{O}(\gamma^S H^S)$, $\gamma = \max\{p_i | 1 \leq i \leq S\}$, which gives the overall time complexity of $\mathcal{O}(\gamma^S H^{2S-2})$. Since $\gamma \leq H$, we deduce that the described DP algorithm requires $\mathcal{O}(H^{3S-2})$ time.

### 3.3. A Branch-and-Bound Algorithm for the BRTP

In this section, we present a B&B algorithm to solve exactly the BRTP. The pseudocode of our B&B algorithm is provided in Algorithm 1 and Appendix A provides additional information concerning

the data structures used in the pseudocode of this algorithm, as well as the rest of the algorithms in the article.

A preprocessing step is performed in the initial bay in order to retrieve all the unobstructed target containers (i.e., in the top tiers). A lower bound on the number of deadlocks for the BRTP is computed as follows. For each stack $i \in \{1, \ldots, S\}$, the lower bound $LB_i$ on the number of relocations needed to retrieve all target containers is $\sum_{k=1}^{p_i} u_{i,k}$, and the global bound is $LB = \sum_{i=1}^{S} LB_i$. In addition, an upper bound on the total number of relocations needed to retrieve all target containers can be set to a high enough value or computed using a linear time algorithm (Algorithm 2 to be presented in Section 3.4), when applicable.

The B&B root node is initialized with the initial configuration of the bay, and the branching strategy consists of enumerating all possible retrieval operations starting from the given bay configuration (and corresponding relocations). Algorithm 1 explores the B&B search tree in a depth-first fashion. At each node of the tree, if the number of relocations executed so far for retrieving the target block plus the lower bound in the number of relocations that will be necessary to retrieve the remaining target containers is greater than or equal the best BRTP upper bound, the node is fathomed (lines 2 to 9). If any target container sits in the bay and no fathoming condition is reached, new nodes are created from a B&B node (lines 10 to 56) by selecting a stack $s \in \{1, \ldots, S\}$ and retrieving target containers which are no longer obstructed (line 11). If such a stack does not exist, a *pickup* stack containing at least one target container is selected (line 37), as well as a *delivery* stack, in which the non target container will be placed (line 44).

Two strategies are employed in order to reduce the number of choices of pickup and delivery stacks. The strategy for choosing the pickup stack consists in using the same stack from the the previous node, provided that there are obstructing containers above the target (line 22). If such stack is not available (e.g., in the beginning of the search or after retrievals), a stack $s \in \{1, \ldots, S\}$ that contains at least one target container is selected. Likewise, the strategy for delivery stacks consists in first relocating to those stacks with no target containers. If no stack meets this condition,

**Algorithm 1:** Branch-and-bound algorithm for the BRTP.

```
1  BRTP_BranchBound(bay_configuration, target_container, upper_bound, blocks_left, pickup_stack, relocations,
   lower_bound, depth)
2  if (relocations + lower_bound ≥ upper_bound) then
3  |  return; // Fathoming
4  end if
5  if (relocations < upper_bound) and (blocks_left = 0) then
6  |  upper_bound = relocations;  // Update best bound
7  |  Store incumbent solution;
8  |  return; // Fathoming
9  end if
10 if (blocks_left > 0) then
11 |  if (there exists a stack s in bay_configuration with a target block available for retrieval) then
12 |  |  Retrieve the target block from s;
13 |  |  BRTP_BranchBound(bay_configuration, target_container, upper_bound, blocks_left − 1, 0, relocations,
   |  |  lower_bound, depth + 1);
14 |  else
15 |  |  free_stack = 0;
16 |  |  for (s = 1, ..., S) do  // Preprocessing: Find a stack s in bay with no target block
17 |  |  |  if (stack s has no target block and is not full) then
18 |  |  |  |  free_stack = s;
19 |  |  |  |  break;
20 |  |  |  end if
21 |  |  end for
22 |  |  if (pickup_stack ≠ 0) then  // Relocate from previous selected stack with deadlocks
23 |  |  |  if (free_stack ≠ 0) then  // Relocate to stack with no target blocks
24 |  |  |  |  Move an obstructing block from pickup_stack to free_stack;
25 |  |  |  |  Update lower_bound;
26 |  |  |  |  BRTP_BranchBound(bay_configuration, target_container, upper_bound, blocks_left, pickup_stack,
   |  |  |  |  relocations + 1, lower_bound, depth + 1);
27 |  |  |  else  // Find a stack s′ in bay with available space
28 |  |  |  |  for (s′ = 1, ..., S) do
29 |  |  |  |  |  if (pickup_stack ≠ s′ and s′ is not full) then
30 |  |  |  |  |  |  Move an obstructing block from pickup_stack to s′;
31 |  |  |  |  |  |  Update lower_bound;
32 |  |  |  |  |  |  BRTP_BranchBound(bay_configuration, target_container, upper_bound, blocks_left,
   |  |  |  |  |  |  pickup_stack, relocations + 1, lower_bound, depth + 1);
33 |  |  |  |  |  end if
34 |  |  |  |  end for
35 |  |  |  end if
36 |  |  else  // Find a stack s in bay with an obstructed target block
37 |  |  |  for (s = 1, ..., S) do
38 |  |  |  |  if (stack s has a target block) then
39 |  |  |  |  |  if (free_stack ≠ 0) then
40 |  |  |  |  |  |  Move an obstructing block from s to free_stack;
41 |  |  |  |  |  |  Update lower_bound;
42 |  |  |  |  |  |  BRTP_BranchBound(bay_configuration, target_container, upper_bound, blocks_left, s,
   |  |  |  |  |  |  relocations + 1, lower_bound, depth + 1);
43 |  |  |  |  |  else
44 |  |  |  |  |  |  for (s′ = 1, ..., S) do  // Find a stack s′ in bay with available space
45 |  |  |  |  |  |  |  if (s ≠ s′ and s′ is not full) then
46 |  |  |  |  |  |  |  |  Move an obstructing block from s to s′;
47 |  |  |  |  |  |  |  |  Update lower_bound;
48 |  |  |  |  |  |  |  |  BRTP_BranchBound(bay_configuration, target_container, upper_bound,
   |  |  |  |  |  |  |  |  blocks_left, s, relocations + 1, lower_bound, depth + 1);
49 |  |  |  |  |  |  |  end if
50 |  |  |  |  |  |  end for
51 |  |  |  |  |  end if
52 |  |  |  |  end if
53 |  |  |  end for
54 |  |  end if
55 |  end if
56 end if
57 return;
```

any other stack $s' \in \{1, \ldots, S\}$ different from the pickup stack is selected (line 28). Once the pair of stacks is chosen, the topmost container from the pickup stack is relocated to the delivery stack. Note that both strategies would be suboptimal for the 2BRTP.

Finally, if the lower bound on the number of relocations is smaller than the current upper bound, and there is no target container left in the bay, then a new best known BRTP solution is found. The incumbent solution is updated and the node is fathomed.

### 3.4. A Linear Time Algorithm for the BRTP with empty slots

In this section, we describe a BRTP linear algorithm that can solve to optimality BRTP instances in which at any step of the algorithm it is always possible to reach the bottommost target container in at least a stack, such as

$$\exists k \in \{1, \ldots, S\} | LB_k \leq \sum_{i=1, i \neq k}^{S} (h_i - \sum_{j=0}^{p_i} (w_{i,j} + u_{i,j})) \tag{2}$$

During our testing, all the instances adapted from the literature and randomly generated (with bay sizes similar to instances for related problems from the literature) proved to be feasible and solvable with the linear time algorithm. We artificially generated instances to test how often Condition 2 is violated and noticed that only unlikely configurations may produce infeasibility. Details will be provided in Section 5, but the linear algorithm proved to be of practical relevance according to our computational tests.

We now present this algorithm for the BRTP, the pseudocode of which is provided in Algorithm 2. The trivial retrievals are performed at first; i.e., the target containers available at the top of the stacks. Next, $LB_i$ is computed for each stack $i \in \{1, \ldots, S\}$, and the stacks that do not hold target containers are marked as *priority stacks*. The remaining stacks containing target containers are sorted with respect to their $LB_i$ in non-decreasing order in $\mathcal{O}(S)$ (linear) time using the Counting Sort algorithm (Cormen et al. 2009). The following steps are repeated until the sorted list is empty. The first stack in the sequence is selected and all obstructing containers positioned over the target containers are relocated to stacks marked as priority target stacks if available, otherwise

the containers are relocated to the next stacks in the sorted sequence. This can be done in $\mathcal{O}(h)$ complexity, at most $\mathcal{O}(S)$ times, which brings the overall complexity to $\mathcal{O}(n)$. After retrieving all the target containers, the selected stack is removed from the sorted sequence and included in the priority target list. A priority stack with minimum height is chosen for the relocation process. The priority target list is implemented as a linked list of linked lists. The main list is sorted in non-decreasing order according to the number of containers in each non-target stack. Each element in the list is a linked list that contains all the non-target stacks with the same amount of containers. Using this data structure, the complexity for inserting new elements is $\mathcal{O}(H)$ (as in lines 11 and 31), and retrieving the less occupied stack and updating the list can be performed in constant time (as in line 23). The complexity of Algorithm 2 is $\mathcal{O}(n)$.

**Algorithm 2:** Retrieve all containers belonging to a given target group.

```
1  Retrieve_TargetBlocks(bay_configuration, target_container)
2  relocations = 0;
3  sorted_StackList = 0;
4  priority_StackList = 0;
5  Retrieve from bay_configuration all target containers not being blocked;
6  for (s = 1, 2, ..., S) do
7      lower_bound = Target_block_LB(bay_configuration[s], target_container);
8      if (lower_bound > 0) then
9          insert(sorted_StackList, s, lower_bound);
10     else
11         insert(priority_StackList, s);
12     end if
13 end for
14 Sort sorted_StackList accordingly to lower_bound in non-decreasing order;
15 while (size(sorted_StackList) ≠ 0) do
16     s = firstElement(sorted_StackList);
17     while (find(bay_configuration[s], target_container)) do
18         slot = size(bay_configuration[s]);
19         if (bay_configuration[s][slot] = target_container) then
20             retrieve(bay_configuration[s][slot]);
21         else
22             if (size(priority_StackList) > 0) then
23                 Relocate bay_configuration[s][slot] to the stack in priority_StackList less occupied;
24             else
25                 Relocate bay_configuration[s][slot] to the next stack in sorted_StackList with available space;
26             end if
27             relocations = relocations + 1;
28         end if
29     end while
30     remove(sorted_StackList, s);
31     insert(priority_StackList, s);
32 end while
33 return relocations;
```

## 4. The Bi-objective Block Retrieval Problem

In this section, we discuss the computational complexity of the 2BRTP and a B&B algorithm for the problem. Additionally, we also describe a beam search algorithm in order to cope with the intrinsic difficulty of the problem.

An instance of the 2BRTP can be described by providing the following information. There are $S \geq 2$ stacks such that stack $i$ has a height of $h_i$ tiers (slots) numbered from bottom to top. Define $H = \max\{h_i | 1 \leq i \leq S\}$, the maximum height. Each stack $i$ can be seen as an array such that in each cell the group $g$, $1 \leq g \leq G$, of the corresponding container is stored.

A target group $t$ is given, and the primary objective is to retrieve all containers of that group with the smallest possible number of relocations. For each group $g \neq t$, a probability $P_g$ that the containers of that group will form the next target is given. The secondary objective function can be written as $\sum_{g \in \{1,...,G\} \setminus t} P_g C_g$, where $C_g$ is the cost of the retrieval of group $g$ from the bay of the configuration obtained after the containers of the primary target group $t$ have been retrieved at the minimum cost $C_t$.

### 4.1. Proof of $\mathcal{NP}$-Hardness for the 2BRTP

PROPOSITION 2. *The 2BRTP is NP-Hard in the strong sense, even if all groups have the same probability to become the next target.*

*Proof.* The 3-Dimensional Matching (3DM) is used for reduction. Given 3 disjoint sets $X, Y$, and $Z$ with $|X| = |Y| = |Z| = n$, and a set of triplets $T \subseteq X \times Y \times Z$, find $T' \subseteq T$ such that each element of $X, Y$, and $Z$ is contained within exactly one triplet in $T'$. Recall that 3DM is $\mathcal{NP}$-Complete in the strong sense (Garey and Johnson 1979).

Given an instance of 3DM with $X = \{1, ..., n\}$, $Y = \{n+1, ..., 2n\}$, and $Z = \{2n+1, ..., 3n\}$, let $T$ consist of triplets $\tau_k$, $1 \leq k \leq |T|$, and let an integer $g$, $1 \leq g \leq 3n$, occur in $\theta_g$ triplets of set $T$. Based on the described instance of 3DM, we construct an instance of the 2BRTP with $3|T| + 2$ stacks, each of height $H = 3n + 3|T| + 1$.

The number of groups $G$ is defined equal to $3n+|T|+1$, and their structure is as follows. The primary target is group 0 which consists of a single container. For each group $g$, $1 \le g \le 3n$, there are $\theta_g + 1$ containers. Group $3n+1$ consists of $3|T|+1$ containers, while each group $g$, $3n+2 \le g \le 3n+|T|+1$, contains $9|T|+9n$ containers. It is equally probable that a group $g$, $1 \le g \le 3n+|T|+1$, is chosen as the next target, i.e., the probability of such a choice is $P = 1/(3n+|T|+1)$ for any $g \neq 0$.

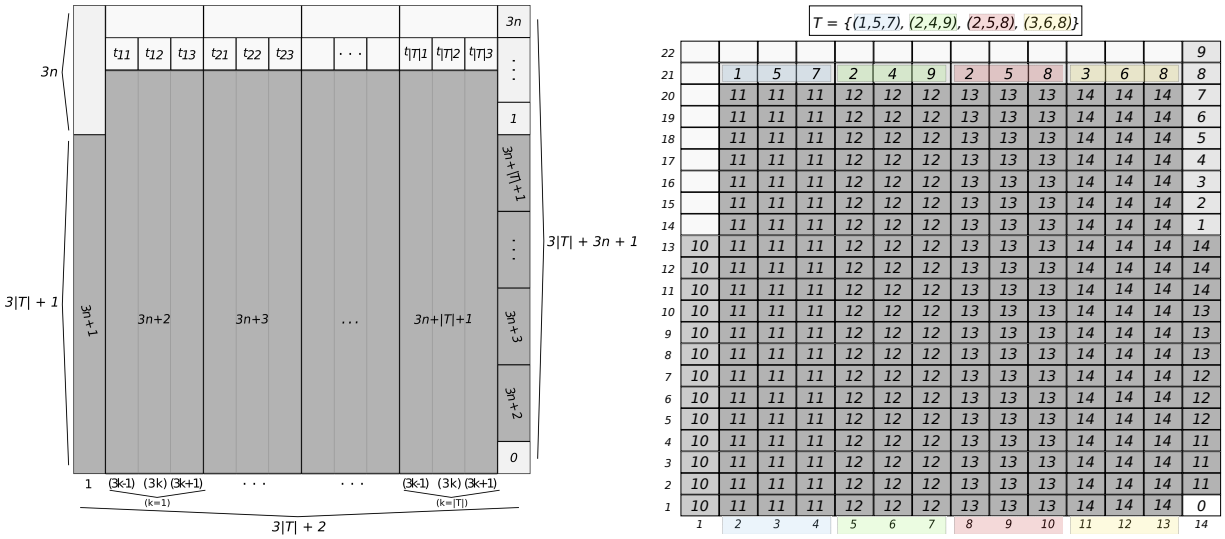
The bay configuration is as follows. The structure of each stack is described from bottom to top. Stack 1 contains $3|T|+1$ containers of group $3n+1$; the remaining slots of that stack are initially free. Each triplet $\tau_k$, $1 \le k \le |T|$, is associated with three stacks $3k-1, 3k$ and $3k+1$. Each of these stacks contains containers of group $3n+k+1$ up to height $3n+3|T|-1$. One container of the group defined by the first, second, and third elements of triple $\tau_k$ is placed on top of stack $3k-1, 3k$, and $3k+1$, respectively. Each of the stacks from 2 to $3|T|+1$ contains one empty slot. The initial configuration of the remaining stack $3|T|+2$ is as follows: it contains one container of the target group 0 at the bottom, then blocks of three containers of the groups $3n+2, 3n+3, \ldots, 3n+|T|+1$ in this order, followed by one container of each group $1, 2, \ldots, 3n$ in this order, leaving no empty slots.

The generic structure of the instance is shown in Figure 5a. For illustration, Figure 5b shows the bay configuration derived for $n = 3$, $X = \{1,2,3\}$, $Y = \{4,5,6\}$, $Z = \{7,8,9\}$, and $T = \{(1,5,7),(2,4,9),(2,5,8),(3,6,8)\}$.

The length of input of 3DM in the unary encoding is $\mathcal{O}(n|T|)$. The total number of slots in the produced bay configuration is $\mathcal{O}\left(n|T|+|T|^2\right) = \mathcal{O}\left(|T|^2\right)$, so that the described input of the 2BRTP can be obtained in time that is polynomial in terms of $\mathcal{O}(n|T|)$, which provides a pseudopolynomial-time reduction.

We prove that 3DM has a solution if and only if for the constructed instance of the 2BRTP the value of the primary objective is at most $3|T|+3n$, while the value of the secondary objective is $P(3n(n+1)/2)+6P|T|$.

**Figure 5** (a) BRTP bay arrangement for generic instances. (b) BRTP bay arrangement for the proposed example.




(a) (b)

Necessity ($\Rightarrow$): Assume that 3DM has a solution given by a collection $T'$ of $n$ triplets. In order to retrieve the container of the target group 0, we need to relocate $3|T|+3n$ other containers from stack $3|T|+2$. Thus, in our solution to the 2BRTP we only may relocate the containers from that stack. We relocate the containers of groups 1 to $3n$ from stack $3|T|+2$ to the top of the stacks corresponding to the triplets in $T'$ to match the container on top. For relocating containers of groups $3n+|T|+1$ down to $3n+2$ from stack $3|T|+2$, we use one of the stacks $2,\ldots,3|T|+1$ for which position $3n+|T|+1$ is empty and the group of the container to be relocated matches that of the container at the bottom of the stack. If such a stack is not available, we relocate the container to the first stack. In the obtained configuration, stack $3|T|+2$ is empty, so that it is possible to retrieve containers of any group that may obstruct a secondary target.

As a result of the relocation process described, the first stack will be totally filled, since $3n$ containers in total will be moved from stack $3|T|+2$. These containers belong to $n$ groups from the set $\{3n+2,...,3n+|T|+1\}$ and will be organized in blocks of three. Thus, a contribution of stack 1 to the secondary objective is $P(3n(n+1)/2)$. We add a single container on top of a stack from 2 to $3|T|+1$ so that its group either matches the group of the containers at the bottom or the

group of the container on top will result in a secondary objective contribution of $2P$, summing up to $6P|T|$ for all such stacks. The value of the secondary function is hence $P\,(3n(n+1)/2)+6P|T|$, as required.

Sufficiency ($\Leftarrow$): Assume that we have a solution to the constructed 2BRTP instance with the values of the primary and the secondary objective functions at most $3|T|+3n$ and $P\,(3n(n+1)/2)+6P|T|$, respectively. As noticed above, the value of the primary objective must be equal to $3|T|+3n$ and no relocations from stacks other than $3\,|T|+2$ are allowed.

The total number of the free slots in the initial configuration is equal to $3|T|+3n$, i.e., in any solution feasible with respect to the primary objective, stack 1 will be filled with $3n$ containers. The contribution of stack 1 to the secondary objective increases as the number of groups in the stack increases. The minimum contribution is realized if, as a result of reallocation, containers that belong to $n$ groups are moved to stack 1 and arranged there as blocks of three for each group. This results into a contribution of $P(3n(n+1)/2)$ from stack 1 to the secondary objective function.

For each stack from the range from 2 to $3|T|+1$, the minimum contribution towards the secondary objective is $2P$, which is realized when the container relocated to this stack either matches the group of the container on top or the group of the containers at the bottom. Relocating a container that does not match the other containers in such a stack will result in a contribution of $3P$. Hence, the minimum total contribution of stacks 2 to $3|T|+1$ is $6P|T|$. The only way of attaining the total minimum of $P\,(3n(n+1)/2)+6P|T|$ is to relocate the containers of group 1 to $3n$ from the last stack the stacks of the range from 2 to $3|T|+1$ that correspond to exactly $n$ triplets in $T$. That means that 3DM must have a solution. $\square$

Notice that Proposition 2 does not resolve the complexity status of the special case of the 2BRTP in which $G=n$, i.e., each group consists of a single container. It can be verified that if the retrieval probabilities for the secondary target are equal for all groups other than the primary target, then the 2BRTP can be solved in polynomial time. However, in the case of unequal probabilities and $G=n$, the complexity status of the 2BRTP is open.

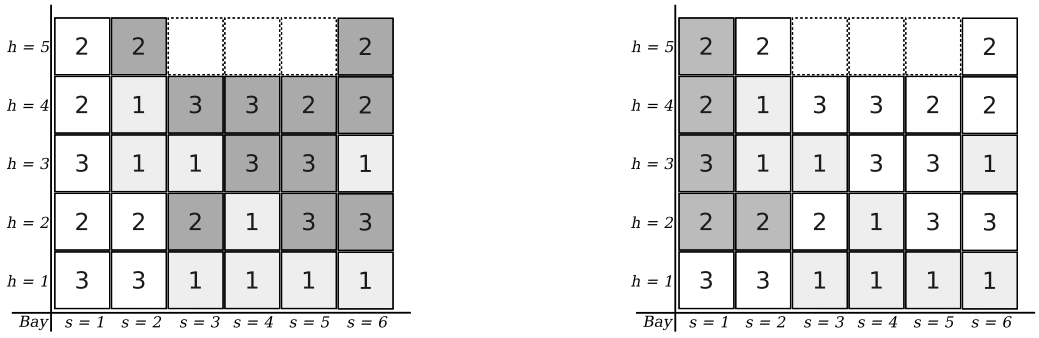
### 4.2. A B&B algorithm for the 2BRTP

The bi-objective structure of the 2BRTP requires the sequential solution of the retrieval of the target container group, as well as all scenarios regarding the retrieval of the remaining groups. Hence, developing an Integer Programming model for 2BRTP is unlikely to yield a tractable formulation. In what follows, we derive a combinatorial bound and a B&B algorithm to solve the 2BRTP.

A lower bound for the secondary 2BRTP objective function approximates the expected number of relocations to retrieve the non-target groups. The bound is based on the concept of counting the obstructing containers above any non-target group and below all target containers in the stack. Figures 6 (a) and (b) depict the container deadlocks that will be taken into account when computing the lower bounds for the primary objective and for the secondary objective, respectively. In Figure 6 (b), the stack $s = 1$ does not contain any target container and thus none of its containers are going to be relocated during the retrieval process of the target group. The same observations can be made about those non-target containers positioned below target containers, as in stack $s = 2$. Nevertheless, once the retrieval process is finished, the containers belonging to group 2 or group 3 will in turn be considered for retrieval and in both cases additional relocations will be necessary. If group 2 is the second target, at least one container of group 3 will be relocated. Similarly, if group 3 is the second target, at least 4 containers of group 2 will be relocated. Denoting the probability of retrieving group $g$ as $P_g$, the secondary objective lower bound for the bay depicted will be $E[Rel] = (1 \times P_2) + (4 \times P_3)$. Containers above any of the target containers cannot be taken into account in the secondary objective computation, given that their relocation may alter the bay configuration.

The pseudocode in Algorithm 3 states the steps required for computing the lower bound for the secondary objective of 2BRTP. For each container group in the bay other than the target, the algorithm counts the number of containers of non target groups located above it in all stacks, if no target container is located in between. These containers will be deadlocks in the secondary objective computation and are therefore added to the lower bound. Note that for each non target

**Figure 6** (a) Target group 1 and respective obstructing containers (in dark gray) which represents the primary objective lower bound, (b) Non-target groups 2 and 3 are used in the secondary objective lower bound.




**Algorithm 3:** Compute the secondary objective function lower bound for a given 2BRTP bay.

```
1  2BRTP_LB(bay_configuration, target_container)
2  lower_bound = 0;
3  for (s = 1, ..., S) do
4      if (size(bay_configuration[s]) > 1) then
5          accounted_Group[1, ..., G] = false;
6          slot = 1;
7          while (slot ≤ size(bay_configuration[s]) − 1) and (bay_configuration[s][slot] ≠ target_container) do
8              aux = slot + 1;
9              while (aux ≤ size(bay_configuration[s])) and (bay_configuration[s][aux] ≠ target_container) do
10                 if (bay_configuration[s][aux] ≠ bay_configuration[s][slot]) then
11                     lower_bound = lower_bound + pr(bay_configuration[s][slot]);
12                     accounted_Group[bay_configuration[s][slot]] = true;
13                 end if
14                 aux = aux + 1;
15             end while
16             repeat
17                 slot = slot + 1;
18             until (accounted_Group[bay_configuration[s][slot]] = true) and (slot ≤ size(bay_configuration[s]));
19         end while
20     end if
21 end for
22 return lower_bound;
```

group and stack, we compute the bound based on the topmost container and stacks containing one container or no containers do not contribute to the lower bound.

We now present our B&B algorithm for the 2BRTP, the pseudocode of which is provided in Algorithm 4. The root node is initialized with the initial configuration of the bay, and the branching strategy consists of enumerating all possible retrieval operations starting from the given bay configuration (and corresponding relocations). Algorithm 4 explores the B&B search tree in a depth first fashion. At each node of the tree, if the number of relocations executed so far for retrieving

the target block is greater than the optimal BRTP solution or the lower bound is greater than or equal to the upper bound on the secondary objective, the node is fathomed.

The branching strategies employed in the BRTP B&B algorithm described in Section 3.3 cannot be applied when solving the 2BRTP, provided they may cut off the optimal solutions. Hence, for the 2BRTP, if any target container sits in the bay and no fathoming condition is reached, new nodes are created from a B&B node by selecting each stack $s \in \{1, \ldots, S\}$ that contains at least one target container. If the target containers in $s$ are not obstructed, they are retrieved. Otherwise, if there are containers obstructing the target containers in the selected stack, any other stack $s' \in \{1, \ldots, S\}, s' \neq s$, is selected and the topmost container from $s$ is relocated to $s'$.

Before executing the B&B, a preprocessing step is performed in the initial bay in order to retrieve all the unobstructed target containers (i.e., in the top tiers). In addition, the optimal number or relocations for the BRTP and the lower and upper bound for the secondary objective are also computed using Algorithms 2 and 3. The upper bound is computed by applying Algorithm 2 for $G-1$ times, each time assuming an alternative $g \in \{1, \ldots, G\} \setminus \{t\}$ being the group to be retrieved second. The overall upper bound is computed by multiplying the number of deadlocks of each group by the probability of retrieval and summing up the contribution of all groups (other than the target).

If the number of relocations is equal to the BRTP optimal solution, the lower bound on the secondary objective is smaller than the current upper bound, and there is no target container left in the bay, then a new best known 2BRTP solution was found. The incumbent solution is updated and the node is fathomed.

### 4.3. A Beam Search algorithm for the 2BRTP

The B&B algorithm described above can be very time consuming depending on the size of the instance being solved. In search for a better balance between solution quality and execution time, we incorporated in Algorithm 4 the two restricted branching rules employed in the BRTP B&B algorithm. These rules transform the B&B algorithm into a beam search algorithm since the optimal

**Algorithm 4:** Branch-and-bound algorithm for the 2BRTP.

```
1  2BRTP_BranchBound(bay_configuration, target_container, BRTP_relocations, upper_bound, blocks_left,
   relocations, bound, depth)
2  if (relocations > BRTP_relocations) or (bound ≥ upper_bound) then
3  |  return; // Fathoming
4  end if
5  if (relocations = BRTP_relocations) and (bound < upper_bound) and (blocks_left = 0) then
6  |  second_bound = 0;
7  |  for (g = 1, ..., G) do  // Compute second objective with BRTP algorithm
8  |  |  if (g ≠ target_container) then
9  |  |  |  second_bound = second_bound + Retrieve_TargetBlocks(bay_configuration, g);
10 |  |  end if
11 |  end for
12 |  if (second_bound < upper_bound) then
13 |  |  upper_bound = second_bound;  // Update best bound
14 |  |  Store incumbent solution;
15 |  end if
16 |  return; // Fathoming
17 end if
18 if (blocks_left > 0) then
19 |  if (there exists a stack s in bay with a target block available for retrieval) then
20 |  |  Retrieve the target block from s;
21 |  |  2BRTP_BranchBound(bay_configuration, target_container, BRTP_relocations, upper_bound, blocks_left − 1,
   |  |  relocations, bound, depth + 1);
22 |  else
23 |  |  for (s = 1, ..., S) do
   |  |  |  // Find a stack s in bay with an obstructed target block
24 |  |  |  if (stack s has a target block) then
25 |  |  |  |  for (s' = 1, ..., S) do
   |  |  |  |  |  // Find a stack s' in bay with available space
26 |  |  |  |  |  if (s ≠ s') then
27 |  |  |  |  |  |  Move an obstructing block from s to s';
28 |  |  |  |  |  |  bound = 2BRTP_LB(bay_configuration[s'], target_container);  // Compute new 2BRTP lower
   |  |  |  |  |  |  bound
29 |  |  |  |  |  |  2BRTP_BranchBound(bay_configuration, target_container, BRTP_relocations,
   |  |  |  |  |  |  upper_bound, blocks_left, relocations + 1, bound, depth + 1);
30 |  |  |  |  |  end if
31 |  |  |  |  end for
32 |  |  |  end if
33 |  |  end for
34 |  end if
35 end if
36 return;
```

solution can be disregarded in the search, but the CPU time requirement decreases significantly. We refer the interested reader to Furcy and Koenig (2005), Zhou and Hansen (2005) for more information about beam search algorithms. We restated those branching strategies in the two following rules:

- *Rule 01 (Cyclic relocations).* If a stack $s$ containing target containers was selected in an iteration and the target containers in $s$ are still obstructed, choose $s$ to relocate another deadlock in the next iteration.

- *Rule 02 (Non-target stacks).* Select a stack $s' \in R$, if $R \neq \emptyset$, to receive a relocated non-target container.

## 5. Computational Experiments

The proposed algorithms were coded in C and executed on an Intel® Core™ i7-2600 3.40 GHz CPU, with 8.0 GB of RAM memory running under GNU/Linux Debian 7.9.

### 5.1. Test Instances

In order to assess the performance of the proposed algorithms, we have transformed the set of BRP instances proposed by Caserta et al. (2012) into BRTP instances. The original BRP set is composed of 840 instances forming 21 classes, each one containing 40 instances. The classes are characterized by the dimensions of the bay ($S \times H$). For each bay, the first $H' = H - 2$ tiers are filled with containers, resulting in a total of $n = S \times H'$ containers. Each one of these $n$ containers is randomly placed in the bay and assigned an identifier in the range 1 to $n$, meaning that each bay is composed by $n$ groups and each group contains a container. The number of stacks is chosen among $S \in \{3, 4, \ldots, 10\}$ and the number of filled tiers among $H' \in \{3, 4, 5, 6, 10\}$. Note that not all combinations of these values were considered in Caserta et al. (2012), in which the authors point out that these bay settings are based on the physical limitations of gantry cranes. The BRP instances were transformed into BRTP instances using the original bay dimensions and amount of occupied tiers per stack. In order to fill the originally occupied space with containers, the following steps are repeated for each stack $s$ until $H'$ tiers have been used. A group $g$, and a number of containers $h'$ smaller than or equal to $H'$ are selected at random and inserted in $s$. If $h' < H'$ the previous step is repeated until the remaining $(H' - h')$ tiers have been filled. Else, the next available stack is processed. Once all stacks are processed in this manner, if there is any group not being used, the whole algorithm is repeated. This procedure was used to generate 2520 instances with $G \in \{3, 4, 5\}$. Note that by construction all instances in this set satisfy the condition to apply Algorithm 2.

We have also developed a random BRTP instance generator which, along with a set of chosen parameters, is used for generating additional instances. The pseudocode for the generator is shown in Algorithm 5. The procedure generates a bay with dimensions ($S \times H$) randomly filled with

containers belonging to $G$ groups. The total amount of containers is proportional to the size of the bay and an occupancy rate parameter. A random group $g \in \{1, \ldots, G\}$ and a random amount $h'$ of containers (smaller than or equal to $H$) belonging to this group are selected and inserted in a stack. These steps are repeated for each stack until they are all completely filled. Next, containers are randomly removed from the bay until the occupancy rate is reached. If the container being removed is in between filled tiers, the topmost containers are moved downwards. Finally, the algorithm halts if at least one container of each group is present in the final bay, otherwise the instance is rejected and the whole process is repeated.

**Algorithm 5:** Generate initial bay configurations randomly.

```
1  Instance_Generator(S, H, G, occ_Rate, seed)
2  counter = 0;
3  Initialize random number generator with seed;
4  number_Containers = ⌊(S × H × occ_Rate)⌋;          // Compute the number of bay slots that will be occupied
5  if (G > number_Containers) then
6  |  G = number_Containers;
7  end if
8  repeat
9  |  for (s = 1, ..., S) do
10 |  |  for (h = 1, ..., H) do
11 |  |  |  bay_configuration[s][h] = 0;
12 |  |  end for
13 |  end for
   |  /* Fill one stack each time with group of containers selected at random */
14 |  for (s = 1, ..., S) do
15 |  |  used_Tiers = 0;
16 |  |  repeat
17 |  |  |  available_Tiers = H − used_Tiers;
18 |  |  |  Select group_id from {1, 2, ..., G} at random;
19 |  |  |  Select new_Containers from {1, 2, ..., available_Tiers} at random;
20 |  |  |  Insert new_Containers of type group_id in bay_configuration[s];
21 |  |  |  used_Tiers = used_Tiers + new_Containers;
22 |  |  until (used_Tiers = H);
23 |  |  Remove ((S × H) − number_Containers) blocks from the bay_configuration at random ;   // Achieve the
   |  |  occupancy rate
24 |  end for
25 until Each group (1, ..., G) appears at least once in the bay_configuration;
26 return bay_configuration;
```

The procedure described above was used to generate 3240 instances with $S \in \{4, 6, 8, 10\}$, $H \in \{4, 5, 6\}$, and $G \in \{3, 4, 5\}$. For each combination of the these three parameters, 30 instances were generated for the occupancy rates of 70%, 75%, and 80%, resulting in 3240 instances. During the experiments, group number 1 was designated to be the target. A CPU time limit of 5 minutes was imposed for each instance. All these instances satisfied the condition to apply Algorithm 2 (we also increased the occupancy rate to 85%, 90% and 95% but the result was the same).

We therefore generated instances with the following artificial characteristics: $S = 10, H \in \{10, 15, 20\}, G = 3$ and occupancy rate 95%. These bay configurations do not respect standard height of stacks in the landside of a container terminal, but this, combined with a very high occupancy rate, decreases the likelihood of finding stacks in which the bottommost target container can be retrieved. Moreover, we slightly modified the instance generator procedure to further increase the chances of producing infeasible instances for the linear algorithm, by imposing in Line 20 that new_Containers $= 1$. This would result in instances in which many pairs are in the same stack. Among 3000 generated instances, 19/22/62 did not respect condition (2) for $H \in \{10, 15, 20\}$, respectively. Among these, all $H \in \{10, 15\}$ were solved in less than one second by the B&B (few were the feasible retrievals in the bay), but one instance that required 57 seconds. The latter's optimal solution was 122 relocations, therefore a configuration that would require a couple of hours for the retrieval of the containers (assuming one relocation might take 1 minute) and therefore not realistic in real world settings. Similarly, the B&B failed to converge within the time limit in the instances with $H = 20$ due to the very large number of relocations.

### 5.2. Results of instances with equal retrieval probabilities

The following column headings are employed in Tables 1-11. Average results are reported for each instance class and set.

- *Instance Set*: specifies the instance set.
- *OccRate*: the occupancy rate employed in the respective instance set.
- *Rel*: the average number of relocations.
- *E[Rel]*: the average expected number of relocations for retrieving the non-target groups
- *Time(s)*: the average computational time in seconds.
- *Gap(%)*: the gap between the average expected number of relocations of non-target containers considering the final configuration obtained at the end of Algorithm 2 and the average expected number of relocations obtained using the B&B algorithm, computed as $Gap(\%) = (100 \times$ (BRTP E[Rel] $-$ B&B E[Rel])/B&B E[Rel]).

- *BGap(%)*: the gap between the *E[Rel]* obtained by the B&B algorithm (B&B E[Rel]) and the *E[Rel]* obtained using one of the beam search algorithms (Beam E[Rel]), computed as $BGap(\%) = (100 \times (\text{B\&B E[Rel]} - \text{Beam E[Rel]})/\text{B\&B E[Rel]})$.
- *Solved*: the number of instances that the relevant algorithm was able to solve to optimality within the time limit.
- *Less 1s*: the number of instances, among the solved ones, that the respective algorithm was able to solve in less than one second.
- *Optimal*: the number of *solved* instances in which the beam search algorithm was able to find the optimal solution obtained using the B&B algorithm.

Before presenting a more detailed analysis of the computational results for the 2BRTP B&B and beam search algorithms, we will analyze the performance of the BRTP linear algorithm and the BRTP B&B algorithm. Table 1 presents the results for the BRTP linear time algorithm and the BRTP B&B algorithm. In terms of computational time, both the BRTP linear time algorithm and the B&B are able to solve all the BRTP instances to optimality in less than one second.

The linear time algorithm and the B&B algorithm can be used to find upper bounds for the 2BRTP, if used sequentially. The algorithms first are employed to generate a bay configuration in which all target containers are retrieved (with minimum cost), and then the same algorithms can be used to compute the cost of retrieval of all other groups.

Table 1 compares the performance of these upper bounds (the one using sequentially the linear time algorithm and the BRTP B&B, respectively). On average, the upper bounds provided by the BRTP linear time algorithm are at least 20% better than those of the BRTP B&B. The better upper bounds have some impact when solving the 2BRTP. The 2BRTP B&B using the BRTP linear time algorithm was able to solve 3 instances more than the 2BRTP B&B using the BRTP B&B within the time limit, and on average they have similar computational times. Provided the good quality of the results obtained by the BRTP linear time algorithm, in the remaining of the computational experiments, we will be using the linear time algorithm algorithm as heuristic 2BRTP solver.

**Table 1** Results of the BRTP linear time and B&B algorithms on equal retrieval probabilities instances.

| | | BRTP linear | | | | BRTP B&B | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance Set | OccRate | Rel | E[Rel] | Solved | Time(s) | Rel | E[Rel] | Rel Gap(%) | Gap(%) | Solved | Time(s) |
| BRTP | 70% | 2.08 | 1.38 | 1077 | 0.37 | 2.08 | 1.66 | 0.00 | -22.18 | 1078 | 0.54 |
| | 75% | 2.40 | 1.55 | 1067 | 0.27 | 2.40 | 1.84 | 0.00 | -22.16 | 1066 | 0.40 |
| | 80% | 2.43 | 1.75 | 1074 | 0.71 | 2.43 | 2.00 | 0.00 | -20.32 | 1073 | 0.50 |
| Caserta et al. (2012) | $60\% - 85\%$ | 3.22 | 2.32 | 2387 | 1.32 | 3.22 | 2.69 | 0.00 | -21.20 | 2385 | 1.28 |
| | | | | 2BRTP B&B | | | | | | 2BRTP B&B | |

Tables 2, 3, 4 present detailed results for the 2BRTP B&B algorithm using the BRTP linear time algorithm in bays of different dimensions, variable number of groups, and occupancy rate of 70%, 75%, and 80%, respectively. The B&B algorithm is capable of solving 1077, 1067, and 1074 out of 1080 instances within the specified time limit, respectively. In addition, 98% of the instances were solved in less than one second.

**Table 2** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances, occupancy rate 70%.

| | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|
| $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| 4 | 4 | 3 | 0.47 | 0.60 | 0.57 | 0.00 | 3.45 | 30 |
| 4 | 4 | 4 | 0.93 | 0.67 | 0.57 | 0.00 | 17.53 | 30 |
| 4 | 4 | 5 | 0.87 | 0.83 | 0.79 | 0.00 | 9.44 | 30 |
| 4 | 5 | 3 | 1.40 | 0.77 | 0.60 | 0.00 | 15.12 | 30 |
| 4 | 5 | 4 | 1.30 | 1.19 | 0.99 | 0.00 | 36.21 | 30 |
| 4 | 5 | 5 | 1.30 | 1.08 | 0.98 | 0.00 | 12.42 | 30 |
| 4 | 6 | 3 | 2.03 | 1.28 | 1.07 | 0.00 | 17.90 | 30 |
| 4 | 6 | 4 | 1.43 | 1.16 | 1.11 | 0.00 | 1.72 | 30 |
| 4 | 6 | 5 | 1.37 | 1.33 | 1.13 | 0.00 | 29.34 | 30 |
| 6 | 4 | 3 | 0.93 | 0.52 | 0.48 | 0.00 | 1.72 | 30 |
| 6 | 4 | 4 | 1.23 | 0.94 | 0.82 | 0.00 | 18.51 | 30 |
| 6 | 4 | 5 | 0.97 | 0.70 | 0.63 | 0.00 | 12.64 | 30 |
| 6 | 5 | 3 | 2.30 | 0.90 | 0.70 | 0.00 | 16.03 | 30 |
| 6 | 5 | 4 | 3.13 | 1.24 | 0.96 | 0.06 | 32.65 | 30 |
| 6 | 5 | 5 | 1.17 | 1.33 | 1.21 | 0.00 | 18.94 | 30 |
| 6 | 6 | 3 | 3.40 | 1.48 | 0.87 | 0.05 | 55.00 | 30 |
| 6 | 6 | 4 | 2.90 | 1.69 | 1.43 | 0.01 | 29.94 | 30 |
| 6 | 6 | 5 | 1.67 | 1.78 | 1.62 | 0.00 | 15.80 | 30 |
| 8 | 4 | 3 | 1.83 | 0.75 | 0.68 | 0.00 | 4.31 | 30 |
| 8 | 4 | 4 | 1.83 | 0.98 | 0.87 | 0.01 | 14.83 | 30 |
| 8 | 4 | 5 | 1.20 | 1.05 | 0.98 | 0.00 | 14.58 | 30 |
| 8 | 5 | 3 | 2.23 | 1.53 | 1.35 | 0.00 | 8.95 | 30 |
| 8 | 5 | 4 | 2.27 | 1.72 | 1.52 | 0.06 | 20.99 | 30 |
| 8 | 5 | 5 | 2.13 | 1.68 | 1.45 | 0.01 | 25.83 | 30 |
| 8 | 6 | 3 | 3.37 | 1.97 | 1.53 | 4.95 | 27.96 | 30 |
| 8 | 6 | 4 | 3.17 | 1.68 | 1.22 | 0.08 | 57.33 | 30 |
| 8 | 6 | 5 | 2.40 | 2.09 | 1.88 | 0.01 | 24.96 | 30 |
| 10 | 4 | 3 | 2.20 | 1.30 | 1.18 | 0.00 | 9.62 | 30 |
| 10 | 4 | 4 | 2.13 | 1.29 | 1.18 | 0.00 | 10.54 | 30 |
| 10 | 4 | 5 | 1.83 | 1.33 | 1.18 | 0.00 | 18.42 | 30 |
| 10 | 5 | 3 | 3.10 | 1.60 | 1.33 | 0.21 | 15.95 | 29 |
| 10 | 5 | 4 | 2.57 | 1.64 | 1.50 | 0.00 | 5.99 | 30 |
| 10 | 5 | 5 | 2.40 | 1.88 | 1.63 | 0.05 | 24.07 | 30 |
| 10 | 6 | 3 | 4.10 | 2.15 | 1.77 | 1.30 | 15.40 | 30 |
| 10 | 6 | 4 | 3.27 | 2.89 | 2.44 | 0.94 | 17.74 | 29 |
| 10 | 6 | 5 | 3.97 | 2.73 | 2.31 | 5.52 | 42.15 | 29 |
| Average: | | | 2.08 | 1.38 | 1.18 | 0.37 | 19.56 | |

**Table 3** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances, occupancy rate 75%.

| | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|
| $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| 4 | 4 | 3 | 1.27 | 0.72 | 0.62 | 0.00 | 8.33 | 30 |
| 4 | 4 | 4 | 1.53 | 0.80 | 0.71 | 0.00 | 7.10 | 30 |
| 4 | 4 | 5 | 1.07 | 0.83 | 0.79 | 0.00 | 8.33 | 30 |
| 4 | 5 | 3 | 2.00 | 0.90 | 0.72 | 0.00 | 18.00 | 30 |
| 4 | 5 | 4 | 1.57 | 1.10 | 0.93 | 0.00 | 27.76 | 30 |
| 4 | 5 | 5 | 1.20 | 1.33 | 1.23 | 0.00 | 14.37 | 30 |
| 4 | 6 | 3 | 1.97 | 1.22 | 0.98 | 0.06 | 13.14 | 30 |
| 4 | 6 | 4 | 2.03 | 1.57 | 1.33 | 0.00 | 28.87 | 30 |
| 4 | 6 | 5 | 1.83 | 1.63 | 1.50 | 0.00 | 15.15 | 30 |
| 6 | 4 | 3 | 1.70 | 0.93 | 0.80 | 0.16 | 11.43 | 30 |
| 6 | 4 | 4 | 1.57 | 1.00 | 0.82 | 0.00 | 33.06 | 30 |
| 6 | 4 | 5 | 1.20 | 0.97 | 0.87 | 0.00 | 7.02 | 30 |
| 6 | 5 | 3 | 1.90 | 0.90 | 0.77 | 0.00 | 9.26 | 30 |
| 6 | 5 | 4 | 1.53 | 1.04 | 0.93 | 0.00 | 17.33 | 30 |
| 6 | 5 | 5 | 1.87 | 1.28 | 1.17 | 0.00 | 14.12 | 30 |
| 6 | 6 | 3 | 2.67 | 1.98 | 1.65 | 0.00 | 32.90 | 30 |
| 6 | 6 | 4 | 2.83 | 1.81 | 1.46 | 1.96 | 28.92 | 30 |
| 6 | 6 | 5 | 2.00 | 1.95 | 1.73 | 0.01 | 20.08 | 30 |
| 8 | 4 | 3 | 2.20 | 0.98 | 0.90 | 0.00 | 9.77 | 30 |
| 8 | 4 | 4 | 1.63 | 1.12 | 1.02 | 0.00 | 14.48 | 30 |
| 8 | 4 | 5 | 1.93 | 1.10 | 0.98 | 0.00 | 8.33 | 30 |
| 8 | 5 | 3 | 3.20 | 1.70 | 1.40 | 0.00 | 17.09 | 28 |
| 8 | 5 | 4 | 2.37 | 1.88 | 1.71 | 0.02 | 16.27 | 30 |
| 8 | 5 | 5 | 1.83 | 1.97 | 1.78 | 0.01 | 13.92 | 30 |
| 8 | 6 | 3 | 4.97 | 2.50 | 1.98 | 0.12 | 29.94 | 28 |
| 8 | 6 | 4 | 3.40 | 2.30 | 1.86 | 0.01 | 36.76 | 28 |
| 8 | 6 | 5 | 2.73 | 2.49 | 2.24 | 0.00 | 23.43 | 30 |
| 10 | 4 | 3 | 2.47 | 1.18 | 1.00 | 0.00 | 17.36 | 30 |
| 10 | 4 | 4 | 2.90 | 1.56 | 1.36 | 0.01 | 14.30 | 30 |
| 10 | 4 | 5 | 2.13 | 1.36 | 1.16 | 0.00 | 21.50 | 30 |
| 10 | 5 | 3 | 3.57 | 1.57 | 1.35 | 2.45 | 16.38 | 29 |
| 10 | 5 | 4 | 3.30 | 2.03 | 1.62 | 0.45 | 32.40 | 30 |
| 10 | 5 | 5 | 3.17 | 1.89 | 1.59 | 0.13 | 23.88 | 29 |
| 10 | 6 | 3 | 5.33 | 2.38 | 1.87 | 0.00 | 36.61 | 27 |
| 10 | 6 | 4 | 4.00 | 2.98 | 2.56 | 3.97 | 22.17 | 29 |
| 10 | 6 | 5 | 3.63 | 2.92 | 2.42 | 0.34 | 29.78 | 29 |
| Average: | | | 2.40 | 1.55 | 1.33 | 0.27 | 19.43 | |

The results of both BRTP and 2BRTP for the instances of Caserta et al. (2012) with different number of groups are presented in Table 5. The B&B method is capable of solving 94% of the instances within the specified time limit, and 92% of these instances were solved in less than one second. The B&B algorithm requires a longer CPU time to prove optimality when the number of relocations increases. Nevertheless, most of the instances with up to 9 stacks and 7 tiers were solved to optimality. The BRTP linear time algorithm requires less than one second of CPU time for each instance.

In summary, the B&B algorithm solved 97% of the instances to optimality. We observe that 2BRTP allows for average savings in the number of expected number of operations at least of 19% across all instances considered, with respect to solutions obtained using BRTP algorithms

sequentially. This highlights the importance of looking ahead and trying to achieve the best possible bay configuration in the process of retrieving the target containers.

**Table 4** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances, occupancy rate 80%.

| | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|
| $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| 4 | 4 | 3 | 1.03 | 0.52 | 0.45 | 0.00 | 0.00 | 30 |
| 4 | 4 | 4 | 0.90 | 1.03 | 0.90 | 0.00 | 10.29 | 30 |
| 4 | 4 | 5 | 1.00 | 0.91 | 0.85 | 0.00 | 10.22 | 30 |
| 4 | 5 | 3 | 2.13 | 1.00 | 0.88 | 0.00 | 8.65 | 30 |
| 4 | 5 | 4 | 1.47 | 1.27 | 1.14 | 0.00 | 7.47 | 30 |
| 4 | 5 | 5 | 1.47 | 1.31 | 1.24 | 0.00 | 17.50 | 30 |
| 4 | 6 | 3 | 1.83 | 1.38 | 1.22 | 0.00 | 15.56 | 30 |
| 4 | 6 | 4 | 2.07 | 1.79 | 1.54 | 0.01 | 29.27 | 30 |
| 4 | 6 | 5 | 1.70 | 1.93 | 1.73 | 0.00 | 17.21 | 30 |
| 6 | 4 | 3 | 2.33 | 1.15 | 0.88 | 0.00 | 30.17 | 30 |
| 6 | 4 | 4 | 1.27 | 1.11 | 0.98 | 0.00 | 7.13 | 30 |
| 6 | 4 | 5 | 1.73 | 1.20 | 1.05 | 0.00 | 23.43 | 30 |
| 6 | 5 | 3 | 2.30 | 1.10 | 0.85 | 0.00 | 33.46 | 30 |
| 6 | 5 | 4 | 2.43 | 1.49 | 1.29 | 0.00 | 22.01 | 29 |
| 6 | 5 | 5 | 2.57 | 2.01 | 1.80 | 0.02 | 16.34 | 30 |
| 6 | 6 | 3 | 3.17 | 1.85 | 1.38 | 0.25 | 42.99 | 30 |
| 6 | 6 | 4 | 3.77 | 2.38 | 1.80 | 10.86 | 58.94 | 29 |
| 6 | 6 | 5 | 2.03 | 2.00 | 1.80 | 0.00 | 17.31 | 30 |
| 8 | 4 | 3 | 1.87 | 1.15 | 0.97 | 0.00 | 27.01 | 30 |
| 8 | 4 | 4 | 1.73 | 1.19 | 1.01 | 0.00 | 27.78 | 30 |
| 8 | 4 | 5 | 1.50 | 1.28 | 1.13 | 0.00 | 22.06 | 30 |
| 8 | 5 | 3 | 3.03 | 1.82 | 1.52 | 0.00 | 22.77 | 30 |
| 8 | 5 | 4 | 2.67 | 2.28 | 1.94 | 0.00 | 22.24 | 30 |
| 8 | 5 | 5 | 1.93 | 1.93 | 1.78 | 0.00 | 15.17 | 30 |
| 8 | 6 | 3 | 3.87 | 2.47 | 1.95 | 2.39 | 23.53 | 30 |
| 8 | 6 | 4 | 2.47 | 2.14 | 1.80 | 0.00 | 43.00 | 30 |
| 8 | 6 | 5 | 3.30 | 2.40 | 2.05 | 0.02 | 33.01 | 29 |
| 10 | 4 | 3 | 2.97 | 1.53 | 1.23 | 0.10 | 16.73 | 30 |
| 10 | 4 | 4 | 2.63 | 1.59 | 1.37 | 0.00 | 24.59 | 30 |
| 10 | 4 | 5 | 1.53 | 1.82 | 1.67 | 0.00 | 11.76 | 30 |
| 10 | 5 | 3 | 4.53 | 2.18 | 1.80 | 2.47 | 36.92 | 29 |
| 10 | 5 | 4 | 2.80 | 2.38 | 2.11 | 1.19 | 19.74 | 30 |
| 10 | 5 | 5 | 2.87 | 2.43 | 2.20 | 0.42 | 13.12 | 30 |
| 10 | 6 | 3 | 4.57 | 3.12 | 2.43 | 6.48 | 43.74 | 30 |
| 10 | 6 | 4 | 3.77 | 2.87 | 2.37 | 0.00 | 37.17 | 29 |
| 10 | 6 | 5 | 4.07 | 2.98 | 2.58 | 1.27 | 27.55 | 29 |
| Average: | | | 2.43 | 1.75 | 1.49 | 0.71 | 23.22 | |

**5.2.1. Results of Beam Search Algorithms** Table 6 presents the average results for three versions of the beam search method described in Section 4.3. The first version applies the first branching rule, the second applies the second rule, and the third employs both rules. The results obtained are compared with those reported in the previous tables for the B&B algorithm. Summing up for the two instance sets, the beam search employing Rule 01 is capable of solving 99% of the instances within the time limit, 98% of those solved in less than 1 second, and for only 5 instances

**Table 5** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances based on Caserta et al. (2012).

| | | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| data3-3-3 | 3 | 5 | 3 | 0.78 | 0.65 | 0.65 | 0.00 | 0.00 | 40 |
| data3-3-4 | 3 | 5 | 4 | 0.83 | 0.90 | 0.80 | 0.00 | 13.21 | 40 |
| data3-3-5 | 3 | 5 | 5 | 0.85 | 0.99 | 0.94 | 0.00 | 2.95 | 40 |
| data3-4-3 | 4 | 5 | 3 | 0.95 | 0.64 | 0.50 | 0.00 | 6.43 | 40 |
| data3-4-4 | 4 | 5 | 4 | 1.08 | 0.78 | 0.70 | 0.00 | 10.00 | 40 |
| data3-4-5 | 4 | 5 | 5 | 1.10 | 0.96 | 0.85 | 0.00 | 23.21 | 40 |
| data4-4-3 | 4 | 6 | 3 | 2.03 | 1.04 | 0.79 | 0.00 | 24.29 | 40 |
| data4-4-4 | 4 | 6 | 4 | 1.20 | 1.31 | 1.18 | 0.00 | 7.54 | 40 |
| data4-4-5 | 4 | 6 | 5 | 1.68 | 1.41 | 1.28 | 0.00 | 12.73 | 40 |
| data5-4-3 | 4 | 7 | 3 | 2.30 | 1.76 | 1.43 | 0.00 | 17.33 | 40 |
| data5-4-4 | 4 | 7 | 4 | 2.15 | 1.78 | 1.52 | 0.00 | 30.36 | 40 |
| data5-4-5 | 4 | 7 | 5 | 2.43 | 1.83 | 1.60 | 0.00 | 14.68 | 40 |
| data3-5-3 | 5 | 5 | 3 | 1.15 | 0.83 | 0.66 | 0.00 | 18.75 | 40 |
| data3-5-4 | 5 | 5 | 4 | 1.03 | 0.99 | 0.93 | 0.00 | 10.21 | 40 |
| data3-5-5 | 5 | 5 | 5 | 1.00 | 0.84 | 0.78 | 0.00 | 10.25 | 40 |
| data4-5-3 | 5 | 6 | 3 | 2.18 | 1.24 | 0.80 | 0.00 | 38.74 | 40 |
| data4-5-4 | 5 | 6 | 4 | 1.85 | 1.47 | 1.27 | 0.00 | 21.49 | 40 |
| data4-5-5 | 5 | 6 | 5 | 1.55 | 1.47 | 1.28 | 0.02 | 18.66 | 40 |
| data5-5-3 | 5 | 7 | 3 | 2.60 | 1.66 | 1.34 | 0.00 | 19.95 | 40 |
| data5-5-4 | 5 | 7 | 4 | 2.53 | 1.74 | 1.49 | 0.00 | 16.90 | 40 |
| data5-5-5 | 5 | 7 | 5 | 2.58 | 1.83 | 1.55 | 4.04 | 18.12 | 40 |
| data3-6-3 | 6 | 5 | 3 | 1.50 | 1.01 | 0.83 | 0.00 | 16.67 | 40 |
| data3-6-4 | 6 | 5 | 4 | 1.28 | 1.01 | 0.92 | 0.00 | 14.32 | 40 |
| data3-6-5 | 6 | 5 | 5 | 0.55 | 0.96 | 0.91 | 0.00 | 4.19 | 40 |
| data4-6-3 | 6 | 6 | 3 | 2.98 | 1.53 | 1.04 | 0.00 | 51.87 | 39 |
| data4-6-4 | 6 | 6 | 4 | 2.35 | 1.28 | 1.00 | 0.12 | 28.96 | 39 |
| data4-6-5 | 6 | 6 | 5 | 2.28 | 1.89 | 1.63 | 6.03 | 28.97 | 40 |
| data5-6-3 | 6 | 7 | 3 | 4.05 | 1.68 | 1.15 | 2.17 | 44.49 | 39 |
| data5-6-4 | 6 | 7 | 4 | 2.78 | 2.04 | 1.66 | 1.07 | 34.00 | 39 |
| data5-6-5 | 6 | 7 | 5 | 2.28 | 2.11 | 1.84 | 1.85 | 25.11 | 40 |
| data6-6-3 | 6 | 8 | 3 | 3.83 | 2.49 | 1.93 | 0.56 | 33.74 | 39 |
| data6-6-4 | 6 | 8 | 4 | 3.48 | 2.73 | 2.31 | 2.34 | 32.19 | 40 |
| data6-6-5 | 6 | 8 | 5 | 2.85 | 2.90 | 2.51 | 2.73 | 20.62 | 40 |
| data10-6-3 | 6 | 12 | 3 | 8.05 | 5.30 | 4.35 | 1.22 | 46.90 | 31 |
| data10-6-4 | 6 | 12 | 4 | 6.38 | 5.35 | 4.75 | 0.49 | 45.33 | 32 |
| data10-6-5 | 6 | 12 | 5 | 4.68 | 5.32 | 4.81 | 0.94 | 16.39 | 34 |
| data3-7-3 | 7 | 5 | 3 | 1.98 | 1.14 | 0.89 | 0.02 | 19.91 | 40 |
| data3-7-4 | 7 | 5 | 4 | 2.05 | 1.48 | 1.22 | 0.00 | 27.84 | 40 |
| data3-7-5 | 7 | 5 | 5 | 1.65 | 1.28 | 1.14 | 0.00 | 12.80 | 40 |
| data4-7-3 | 7 | 6 | 3 | 3.18 | 1.63 | 1.21 | 0.00 | 40.00 | 40 |
| data4-7-4 | 7 | 6 | 4 | 3.03 | 1.83 | 1.53 | 0.02 | 33.86 | 39 |
| data4-7-5 | 7 | 6 | 5 | 2.33 | 2.00 | 1.71 | 0.22 | 21.32 | 40 |
| data5-7-3 | 7 | 7 | 3 | 3.08 | 1.69 | 1.53 | 0.07 | 13.54 | 38 |
| data5-7-4 | 7 | 7 | 4 | 3.65 | 2.49 | 2.05 | 0.07 | 37.62 | 35 |
| data5-7-5 | 7 | 7 | 5 | 2.80 | 2.41 | 2.09 | 0.29 | 24.06 | 40 |
| data3-8-3 | 8 | 5 | 3 | 1.88 | 0.96 | 0.86 | 0.00 | 8.76 | 40 |
| data3-8-4 | 8 | 5 | 4 | 1.85 | 1.25 | 1.09 | 0.00 | 13.83 | 40 |
| data3-8-5 | 8 | 5 | 5 | 1.48 | 1.25 | 1.15 | 0.00 | 11.44 | 40 |
| data5-8-3 | 8 | 7 | 3 | 4.70 | 2.30 | 1.70 | 0.74 | 42.43 | 39 |
| data5-8-4 | 8 | 7 | 4 | 3.53 | 2.84 | 2.32 | 0.71 | 34.94 | 40 |
| data5-8-5 | 8 | 7 | 5 | 2.98 | 2.68 | 2.30 | 2.01 | 22.70 | 39 |
| data5-9-3 | 9 | 7 | 3 | 4.78 | 2.33 | 1.85 | 0.14 | 38.87 | 35 |
| data5-9-4 | 9 | 7 | 4 | 3.70 | 3.41 | 2.83 | 0.66 | 28.10 | 39 |
| data5-9-5 | 9 | 7 | 5 | 3.95 | 3.43 | 2.88 | 0.23 | 25.39 | 38 |
| data5-10-3 | 10 | 7 | 3 | 6.23 | 2.99 | 2.26 | 12.15 | 56.05 | 33 |
| data5-10-4 | 10 | 7 | 4 | 5.20 | 2.95 | 2.45 | 3.09 | 32.21 | 36 |
| data5-10-5 | 10 | 7 | 5 | 4.53 | 2.99 | 2.68 | 0.00 | 17.63 | 36 |
| data6-10-3 | 10 | 8 | 3 | 6.40 | 4.30 | 3.25 | 0.03 | 47.36 | 35 |
| data6-10-4 | 10 | 8 | 4 | 5.98 | 4.46 | 3.72 | 1.83 | 26.31 | 32 |
| data6-10-5 | 10 | 8 | 5 | 4.93 | 3.93 | 3.24 | 0.09 | 30.63 | 36 |
| data10-10-3 | 10 | 12 | 3 | 12.08 | 6.85 | 5.26 | 21.60 | 68.92 | 24 |
| data10-10-4 | 10 | 12 | 4 | 11.98 | 9.88 | 8.47 | 0.01 | 20.20 | 17 |
| data10-10-5 | 10 | 12 | 5 | 9.85 | 8.01 | 7.03 | 15.43 | 20.73 | 24 |
| Average: | | | | 3.22 | 2.32 | 1.95 | 1.32 | 24.71 | |

the method was not able to find the known optimal solutions. The beam search employing Rule 02 was not capable of solving only seven instances within the time limit, 99% were solved in less than 1 second, and all the known optimal solutions were found. The beam search employing both rules was capable of solving all but 2 instances within the time limit, and only 4 instances were not solved in less than 1 second. The method successfully found the known optimal solutions for all instances but 5, similar to the method using Rule 01. The reason is that the third method inherits the limitations of both rules. Among the three, the method combining the two rules performs better both in terms of computing time and solution quality.

**Table 6** Results of the Beam Search on instances with equal retrieval probabilities.

| Beam Seach - Rule 01 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Instance Set | OccRate | E[Rel] | Time(s) | BGap(%) | Solved | Optimal | Less 1s |
| BRTP | 70% | 1.18 | 0.10 | 0.01 | 1080 | 1076 | 1078 |
| | 75% | 1.32 | 0.02 | 0.35 | 1079 | 1065 | 1075 |
| | 80% | 1.49 | 0.00 | 0.22 | 1080 | 1072 | 1078 |
| Caserta et al. (2012) | 60% – 85% | 1.91 | 0.99 | 1.12 | 2467 | 2387 | 2417 |
| Beam Seach - Rule 02 | | | | | | | |
| Instance Set | OccRate | E[Rel] | Time(s) | BGap(%) | Solved | Optimal | Less 1s |
| BRTP | 70% | 1.18 | 0.00 | 0.06 | 1080 | 1077 | 1080 |
| | 75% | 1.32 | 0.00 | 0.45 | 1080 | 1067 | 1080 |
| | 80% | 1.49 | 0.28 | 0.25 | 1080 | 1074 | 1078 |
| Caserta et al. (2012) | 60% – 85% | 1.86 | 0.28 | 1.89 | 2513 | 2387 | 2499 |
| Beam Seach - Rules 01 and 02 | | | | | | | |
| Instance Set | OccRate | E[Rel] | Time(s) | BGap(%) | Solved | Optimal | Less 1s |
| BRTP | 70% | 1.18 | 0.00 | 0.01 | 1080 | 1076 | 1080 |
| | 75% | 1.32 | 0.00 | 0.39 | 1080 | 1065 | 1080 |
| | 80% | 1.49 | 0.00 | 0.22 | 1080 | 1072 | 1080 |
| Caserta et al. (2012) | 60% – 85% | 1.86 | 0.01 | 1.95 | 2518 | 2387 | 2516 |

### 5.3. Results for instances with unequal retrieval probabilities

The results obtained with the BRTP linear time algorithm and the 2BRTP B&B algorithms for the generated instances with different retrieval probabilities for the groups, and occupancy rates of 70%, 75%, and 80% are similar to those obtained when using equal retrieval probabilities. They are presented in Tables 7, 8, 9 of Appendix B. The B&B algorithm is capable of solving 1076, 1065, and 1073 out of 1080 instances within the specified time limit, respectively. In addition, 98% of the instances were solved in less than one second.

The results of both BRTP and 2BRTP for the instances of Caserta et al. (2012) with different number of groups and different retrieval probabilities per group are presented in Table 10 of

Appendix B. The B&B method is capable of solving 94% of the instances within 5 CPU minutes, and 91% of those were solved in less than one second. Similar to the case with equal retrieval probabilities, the average number of expected relocations, when solving them as a 2BRTP, was reduced by 21% across all instances considered, with respect to the BRTP optimal solutions.

The average results for the three versions of the beam search method are summarized in Table 11 of the appendix. Among the three methods, the beam search combining the two rules outperforms the other two both in terms of time and solution quality.

## 6. Concluding Remarks

In this paper, we study two problems related to the optimal container retrieval, the BRTP and the 2BRTP. We show that both problems are in general NP-hard, however, the BRTP can be solved in polynomial time, provided that the number of stacks is fixed. For the BRTP, a B&B algorithm is designed, as well as and a linear time algorithm that is able to handle instances of practical relevance. For the 2BRTP, a B&B algorithm and a beam search algorithm are developed.

The effectiveness of the 2BRTP B&B and beam search algorithms has been assessed by extensive computational experiments on two benchmark sets with instance bay sizes varying from small to large dimensions. The B&B algorithm is capable of solving 97% of the instances to optimality within 5 CPU minutes, and the average number of expected relocations has been reduced by at least 21% with respect to the BRTP optimal solutions. Three versions of the beam search algorithm that employ different branch rules have been tested. The most efficient version is capable of solving 99.9% of the instances within the time limit. The reduction in the average number of expected relocations varies among 1% and 2% when compared to the results obtained by the B&B algorithm. The 2BRTP instances with up to 10 stacks in a bay with a maximum height of 6 and 5 groups of customers can be solved consistently to optimality within 5 minutes of CPU time.

Further research may focus on multiple bays, and optimizing the expected number of relocations when more information about the sequence of retrieval for the non-target containers is available.

## Appendix

### A. Pseudocode Notation and Data Structures

In the pseudocodes shown in Algorithms 1, 2, 3, 4, and 5, the following data structures, functions, and variables are employed.

- *accounted_Group* is an array of size $G$. It contains the information regarding if a given group appears or not in a stack.
- *aux* is an auxiliary variable used to identify a specific slot of a stack or array.
- *available_Tiers* is a variable that contains the number of free tiers/slots in a stack.
- *bay_configuration* is a matrix of dimensions $(S \times H)$. It holds the current configuration of the container bay.
- *bound* is a variable that contains the value of the bound of the incumbent 2BRTP solution.
- *blocks_left* is a variable that stores how many target containers are still in the bay and need to be retrieved.
- *BRTP_relocations* is a variable that contains the number of relocations need retrieve all the target containers from a given bay (i.e., the BRTP solution).
- *depth* is a variable that stores the level of the branch-and-bound tree being explored.
- *find*() is a function that informs if a given stack contains a given target container.
- *first_Element*() is a function that informs the first valid element from an array.
- *free_stack* is a variable used to identify a stack that does not contain target blocks and has empty slots.
- *group_id* is a variable used to identify the group for which containers will be generated.
- *insert*() is a function used to insert new elements in the arrays *sorted_StackList* and *priority_StackList*. The new elements are inserted after the last used position.
- *lower_bound* is a variable that stores the value of the best lower bound.
- *new_Containers* is a variable that informs the amount of containers that will be inserted for the selected group.
- *number_Container* is a variable that informs the number of containers to be inserted in the bay being generated.
- *occ_Rate* is a variable that stores the occupancy rate of a bay.
- *pickup_stack* is a variable that keep track of the stack from where non target containers have to be relocated in order to access the target containers being obstructed.
- *pr*() is a function that informs the probability of a given container group.
- *priority_StackList* is an array of size $S$. It stores the information about which stacks does not contain target containers and has available slots.

• *relocations* is a variable used to count the number of relocations performed during the retrieval process performed by the algorithms.

• *remove*() is a function used to remove an element from the array specified.

• *retrieve*() is a function used to retrieved a target container from the bay slot specified by the parameters. It updates the affected data structures; e.g., $bay\_configuration$.

• *second_bound* is an auxiliary variable employed during the computation of the secondary objective in a 2BRTP solution.

• *seed* is a variable used to initialize the random number generator.

• *size*() is a function that informs the size of an array; i.e., it informs the number of elements that an array contains.

• *slot* is a variable used to identify a specific slot of a stack or array.

• *sorted_StackList* is an array of size $S$. It stores the information about the stacks that contains target containers and it is sorted according to the number of deadlock in each stack.

• $Target\_block\_LB()$ is a function that computes the number of deadlocks for a given bay and target group.

• *target_container* is a variable that keep track of the current target group.

• *upper_bound* is a variable that stores the value of the best upper bound.

• $used\_Tiers$ is a variable that contains the number of occupied tiers/slots in a stack.

### B. Results for instances with unequal retrieval probabilities

During our experiments with unequal retrieval probabilities, we generate the probabilities $P_g$, for each group $g \in \{1, \ldots, G\}$, $g \neq t$ as follows. For the first group, the probability is set to be a random number in $r_1 = ]0.0, 1[$, for the second group the probability is a random number in $]0.0, 1 - r_1[$, and we similarly assign the probabilities to the other groups, but the last. For the latter, the probability is set to $1 - \sum_{i=1,\ldots,G-1} r_i$.

Tables 7, 8, 9 present the B&B results of both BRTP and 2BRTP for the generated instances with different retrieval probabilities for the groups, and occupancy rates of 70%, 75%, and 80%.

The results of both BRTP and 2BRTP for the instances of Caserta et al. (2012) with different number of groups and different retrieval probabilities per group are presented in Table 10. The average results for the three versions of the beam search method are summarized in Table 11.

## References

Borjian S, Galle V, Manshadi VH, Barnhart C, Jaillet P (2015a) Container relocation problem: Approximation, asymptotic, and incomplete information. *CoRR* abs/1505.04229, URL `http://arxiv.org/abs/1505.04229`.

**Table 7** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances, occupancy rate 70%, and unequal retrieval probabilities.

| | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|
| $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| 4 | 4 | 3 | 0.47 | 0.60 | 0.58 | 0.00 | 1.79 | 30 |
| 4 | 4 | 4 | 0.93 | 0.69 | 0.58 | 0.00 | 17.55 | 30 |
| 4 | 4 | 5 | 0.87 | 0.79 | 0.73 | 0.00 | 15.99 | 30 |
| 4 | 5 | 3 | 1.40 | 0.76 | 0.58 | 0.00 | 16.77 | 30 |
| 4 | 5 | 4 | 1.30 | 1.18 | 0.97 | 0.00 | 39.42 | 30 |
| 4 | 5 | 5 | 1.30 | 1.14 | 1.04 | 0.00 | 10.39 | 30 |
| 4 | 6 | 3 | 2.03 | 1.31 | 1.13 | 0.00 | 13.90 | 30 |
| 4 | 6 | 4 | 1.43 | 1.14 | 1.08 | 0.00 | 3.53 | 30 |
| 4 | 6 | 5 | 1.37 | 1.30 | 1.09 | 0.00 | 30.87 | 30 |
| 6 | 4 | 3 | 0.93 | 0.49 | 0.45 | 0.00 | 2.27 | 30 |
| 6 | 4 | 4 | 1.23 | 0.91 | 0.79 | 0.00 | 18.72 | 30 |
| 6 | 4 | 5 | 0.97 | 0.74 | 0.65 | 0.00 | 14.64 | 30 |
| 6 | 5 | 3 | 2.30 | 0.85 | 0.68 | 0.00 | 11.89 | 30 |
| 6 | 5 | 4 | 3.13 | 1.24 | 0.96 | 0.08 | 29.09 | 30 |
| 6 | 5 | 5 | 1.17 | 1.36 | 1.23 | 0.00 | 17.86 | 30 |
| 6 | 6 | 3 | 3.40 | 1.45 | 0.81 | 0.06 | 64.82 | 30 |
| 6 | 6 | 4 | 2.90 | 1.70 | 1.43 | 0.01 | 32.70 | 30 |
| 6 | 6 | 5 | 1.67 | 1.92 | 1.74 | 0.00 | 20.85 | 30 |
| 8 | 4 | 3 | 1.83 | 0.65 | 0.58 | 0.00 | 4.31 | 30 |
| 8 | 4 | 4 | 1.83 | 0.96 | 0.85 | 0.01 | 14.90 | 30 |
| 8 | 4 | 5 | 1.20 | 1.02 | 0.94 | 0.00 | 23.73 | 30 |
| 8 | 5 | 3 | 2.23 | 1.61 | 1.41 | 0.00 | 8.81 | 30 |
| 8 | 5 | 4 | 2.27 | 1.71 | 1.50 | 0.10 | 22.76 | 30 |
| 8 | 5 | 5 | 2.13 | 1.68 | 1.45 | 0.02 | 25.23 | 30 |
| 8 | 6 | 3 | 3.37 | 2.09 | 1.66 | 4.87 | 31.94 | 30 |
| 8 | 6 | 4 | 3.17 | 1.71 | 1.24 | 0.09 | 63.24 | 30 |
| 8 | 6 | 5 | 2.40 | 2.01 | 1.78 | 0.01 | 42.63 | 30 |
| 10 | 4 | 3 | 2.20 | 1.36 | 1.24 | 0.00 | 8.38 | 30 |
| 10 | 4 | 4 | 2.13 | 1.27 | 1.17 | 0.00 | 9.94 | 30 |
| 10 | 4 | 5 | 1.83 | 1.33 | 1.18 | 0.00 | 20.29 | 30 |
| 10 | 5 | 3 | 3.10 | 1.62 | 1.32 | 0.20 | 20.12 | 29 |
| 10 | 5 | 4 | 2.57 | 1.62 | 1.48 | 0.00 | 5.41 | 30 |
| 10 | 5 | 5 | 2.40 | 1.97 | 1.72 | 0.17 | 23.40 | 30 |
| 10 | 6 | 3 | 4.10 | 2.13 | 1.75 | 1.91 | 15.41 | 30 |
| 10 | 6 | 4 | 3.27 | 2.91 | 2.47 | 1.07 | 18.52 | 29 |
| 10 | 6 | 5 | 3.97 | 2.60 | 2.21 | 0.09 | 52.61 | 28 |
| Average: | | | 2.08 | 1.38 | 1.18 | 0.24 | 21.52 | |

Borjian S, Manshadi VH, Barnhart C, Jaillet P (2015b) Managing relocation and delay in container terminals with flexible service policies. *CoRR* abs/1503.01535, URL `http://arxiv.org/abs/1503.01535`.

Carlo HJ, Vis IFA, Roodbergen KJ (2013) Seaside operations in container terminals: literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal* 27(2):224–262.

Carlo HJ, Vis IFA, Roodbergen KJ (2014a) Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research* 235(2):412–430.

Carlo HJ, Vis IFA, Roodbergen KJ (2014b) Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research* 236(1):1–13.

**Table 8** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances, occupancy rate 75%, and unequal retrieval probabilities.

| | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|
| $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| 4 | 4 | 3 | 1.27 | 0.71 | 0.61 | 0.00 | 10.49 | 30 |
| 4 | 4 | 4 | 1.53 | 0.80 | 0.70 | 0.00 | 8.13 | 30 |
| 4 | 4 | 5 | 1.07 | 0.86 | 0.80 | 0.00 | 11.82 | 30 |
| 4 | 5 | 3 | 2.00 | 0.91 | 0.71 | 0.00 | 22.18 | 30 |
| 4 | 5 | 4 | 1.57 | 1.12 | 0.94 | 0.00 | 30.67 | 30 |
| 4 | 5 | 5 | 1.20 | 1.39 | 1.27 | 0.00 | 13.97 | 30 |
| 4 | 6 | 3 | 1.97 | 1.13 | 0.90 | 0.07 | 11.62 | 30 |
| 4 | 6 | 4 | 2.03 | 1.49 | 1.26 | 0.00 | 30.50 | 30 |
| 4 | 6 | 5 | 1.83 | 1.58 | 1.44 | 0.00 | 24.94 | 30 |
| 6 | 4 | 3 | 1.70 | 0.92 | 0.77 | 0.19 | 15.39 | 30 |
| 6 | 4 | 4 | 1.57 | 0.99 | 0.80 | 0.00 | 34.06 | 30 |
| 6 | 4 | 5 | 1.20 | 0.96 | 0.84 | 0.00 | 10.86 | 30 |
| 6 | 5 | 3 | 1.90 | 0.95 | 0.84 | 0.00 | 9.22 | 30 |
| 6 | 5 | 4 | 1.53 | 1.06 | 0.92 | 0.00 | 24.24 | 30 |
| 6 | 5 | 5 | 1.87 | 1.36 | 1.23 | 0.00 | 16.47 | 30 |
| 6 | 6 | 3 | 2.67 | 2.05 | 1.75 | 0.00 | 28.16 | 30 |
| 6 | 6 | 4 | 2.83 | 1.83 | 1.44 | 2.28 | 32.15 | 30 |
| 6 | 6 | 5 | 2.00 | 2.01 | 1.76 | 0.02 | 19.66 | 30 |
| 8 | 4 | 3 | 2.20 | 1.02 | 0.93 | 0.00 | 13.11 | 30 |
| 8 | 4 | 4 | 1.63 | 1.12 | 1.02 | 0.00 | 13.37 | 30 |
| 8 | 4 | 5 | 1.93 | 1.10 | 0.98 | 0.00 | 9.23 | 30 |
| 8 | 5 | 3 | 3.20 | 1.74 | 1.44 | 0.00 | 18.21 | 28 |
| 8 | 5 | 4 | 2.37 | 1.96 | 1.79 | 0.00 | 18.69 | 30 |
| 8 | 5 | 5 | 1.83 | 1.97 | 1.79 | 0.01 | 13.32 | 30 |
| 8 | 6 | 3 | 4.97 | 2.38 | 1.85 | 0.37 | 35.20 | 27 |
| 8 | 6 | 4 | 3.40 | 2.32 | 1.86 | 0.01 | 37.34 | 28 |
| 8 | 6 | 5 | 2.73 | 2.50 | 2.22 | 0.00 | 23.64 | 30 |
| 10 | 4 | 3 | 2.47 | 1.25 | 1.05 | 0.00 | 21.50 | 30 |
| 10 | 4 | 4 | 2.90 | 1.59 | 1.40 | 0.03 | 14.56 | 30 |
| 10 | 4 | 5 | 2.13 | 1.37 | 1.17 | 0.00 | 23.28 | 30 |
| 10 | 5 | 3 | 3.57 | 1.62 | 1.36 | 2.66 | 20.85 | 29 |
| 10 | 5 | 4 | 3.30 | 2.03 | 1.60 | 0.36 | 37.54 | 30 |
| 10 | 5 | 5 | 3.17 | 1.80 | 1.50 | 0.24 | 30.60 | 29 |
| 10 | 6 | 3 | 5.33 | 2.37 | 1.90 | 0.00 | 37.72 | 27 |
| 10 | 6 | 4 | 4.00 | 2.93 | 2.48 | 0.01 | 21.07 | 28 |
| 10 | 6 | 5 | 3.63 | 2.82 | 2.30 | 0.55 | 34.80 | 29 |
| Average: | | | 2.40 | 1.56 | 1.32 | 0.19 | 21.63 | |

Caserta M, Schwarze S, Voß S (2011) Container rehandling at maritime container terminals. Böse JW, ed., *Handbook of Terminal Planning*, volume 49 of *Operations Research/Computer Science Interfaces Series*, 247–269 (Springer New York).

Caserta M, Schwarze S, Voß S (2012) A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research* 219(1):96–104.

Caserta M, Voß S (2009) A corridor method-based algorithm for the pre-marshalling problem. Giacobini M, Brabazon A, Cagnoni S, Caro GAD, Ekárt A, Esparcia-Alcázar AI, Farooq M, Fink A, Machado P, eds., *Applications of Evolutionary Computing*, volume 5484 of *Lecture Notes in Computer Science*, 788–797 (Springer Berlin Heidelberg).

**Table 9** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances, occupancy rate 80%, and unequal retrieval probabilities.

| | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|
| $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| 4 | 4 | 3 | 1.03 | 0.52 | 0.46 | 0.00 | 0.00 | 30 |
| 4 | 4 | 4 | 0.90 | 1.03 | 0.89 | 0.00 | 10.45 | 30 |
| 4 | 4 | 5 | 1.00 | 0.86 | 0.78 | 0.00 | 25.60 | 30 |
| 4 | 5 | 3 | 2.13 | 1.12 | 0.99 | 0.00 | 14.85 | 30 |
| 4 | 5 | 4 | 1.47 | 1.27 | 1.15 | 0.00 | 7.83 | 30 |
| 4 | 5 | 5 | 1.47 | 1.29 | 1.21 | 0.00 | 40.67 | 30 |
| 4 | 6 | 3 | 1.83 | 1.42 | 1.26 | 0.00 | 13.36 | 30 |
| 4 | 6 | 4 | 2.07 | 1.76 | 1.50 | 0.01 | 38.67 | 30 |
| 4 | 6 | 5 | 1.70 | 1.92 | 1.73 | 0.00 | 16.00 | 30 |
| 6 | 4 | 3 | 2.33 | 1.19 | 0.91 | 0.00 | 29.19 | 30 |
| 6 | 4 | 4 | 1.27 | 1.12 | 0.97 | 0.00 | 9.70 | 30 |
| 6 | 4 | 5 | 1.73 | 1.25 | 1.07 | 0.00 | 31.60 | 30 |
| 6 | 5 | 3 | 2.30 | 1.17 | 0.91 | 0.00 | 39.45 | 30 |
| 6 | 5 | 4 | 2.43 | 1.47 | 1.25 | 0.00 | 23.58 | 29 |
| 6 | 5 | 5 | 2.57 | 1.97 | 1.77 | 0.02 | 19.23 | 30 |
| 6 | 6 | 3 | 3.17 | 1.69 | 1.27 | 0.26 | 35.03 | 30 |
| 6 | 6 | 4 | 3.77 | 2.42 | 1.82 | 1.24 | 61.73 | 28 |
| 6 | 6 | 5 | 2.03 | 2.11 | 1.85 | 0.00 | 20.08 | 30 |
| 8 | 4 | 3 | 1.87 | 1.18 | 0.98 | 0.00 | 27.54 | 30 |
| 8 | 4 | 4 | 1.73 | 1.21 | 1.05 | 0.00 | 25.63 | 30 |
| 8 | 4 | 5 | 1.50 | 1.28 | 1.11 | 0.00 | 28.92 | 30 |
| 8 | 5 | 3 | 3.03 | 1.79 | 1.48 | 0.00 | 27.70 | 30 |
| 8 | 5 | 4 | 2.67 | 2.33 | 2.00 | 0.00 | 19.68 | 30 |
| 8 | 5 | 5 | 1.93 | 1.96 | 1.80 | 0.00 | 18.13 | 30 |
| 8 | 6 | 3 | 3.87 | 2.41 | 1.88 | 4.42 | 26.87 | 30 |
| 8 | 6 | 4 | 2.47 | 2.16 | 1.78 | 0.00 | 43.75 | 30 |
| 8 | 6 | 5 | 3.30 | 2.43 | 2.07 | 0.05 | 35.14 | 29 |
| 10 | 4 | 3 | 2.97 | 1.64 | 1.30 | 0.10 | 17.30 | 30 |
| 10 | 4 | 4 | 2.63 | 1.62 | 1.38 | 0.00 | 26.02 | 30 |
| 10 | 4 | 5 | 1.53 | 1.88 | 1.73 | 0.00 | 12.42 | 30 |
| 10 | 5 | 3 | 4.53 | 2.36 | 1.95 | 2.84 | 35.98 | 29 |
| 10 | 5 | 4 | 2.80 | 2.36 | 2.09 | 1.33 | 19.41 | 30 |
| 10 | 5 | 5 | 2.87 | 2.40 | 2.15 | 0.11 | 16.47 | 30 |
| 10 | 6 | 3 | 4.57 | 3.16 | 2.46 | 6.92 | 57.97 | 30 |
| 10 | 6 | 4 | 3.77 | 2.93 | 2.40 | 0.65 | 35.75 | 29 |
| 10 | 6 | 5 | 4.07 | 2.91 | 2.51 | 1.81 | 26.26 | 29 |
| Average: | | | 2.43 | 1.77 | 1.50 | 0.55 | 26.05 | |

Castilho B, Daganzo CF (1993) Handling strategies for import containers at marine terminals. *Transportation Research Part B: Methodological* 27(2):151–166.

Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) *Introduction to Algorithms, Third Edition* (Cambridge, Massachusetts, USA: The MIT Press), 3rd edition.

Dekker R, Voogd P, van Asperen E (2006) Advanced methods for container stacking. *OR Spectrum* 28(4):563–586.

Flajolet P, Sedgewick R (2009) *Analytic Combinatorics* (New York, NY, USA: Cambridge University Press), 1 edition.

**Table 10** Results of the BRTP linear time and 2BRTP B&B algorithms for 2BRTP instances based on Caserta et al. (2012), and unequal retrieval probabilities.

| | | | | BRTP linear | | 2BRTP B&B | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | $S$ | $H$ | $G$ | Rel | E[Rel] | E[Rel] | Time(s) | Gap(%) | Solved |
| data3-3-3 | 3 | 5 | 3 | 0.78 | 0.73 | 0.73 | 0.00 | 0.00 | 40 |
| data3-3-4 | 3 | 5 | 4 | 0.83 | 0.89 | 0.79 | 0.00 | 13.13 | 40 |
| data3-3-5 | 3 | 5 | 5 | 0.85 | 0.95 | 0.91 | 0.00 | 2.92 | 40 |
| data3-4-3 | 4 | 5 | 3 | 0.95 | 0.62 | 0.48 | 0.00 | 6.83 | 40 |
| data3-4-4 | 4 | 5 | 4 | 1.08 | 0.77 | 0.68 | 0.00 | 12.00 | 40 |
| data3-4-5 | 4 | 5 | 5 | 1.10 | 0.94 | 0.83 | 0.00 | 33.55 | 40 |
| data4-4-3 | 4 | 6 | 3 | 2.03 | 1.07 | 0.81 | 0.00 | 31.93 | 40 |
| data4-4-4 | 4 | 6 | 4 | 1.20 | 1.26 | 1.13 | 0.00 | 10.49 | 40 |
| data4-4-5 | 4 | 6 | 5 | 1.68 | 1.37 | 1.21 | 0.00 | 21.94 | 40 |
| data5-4-3 | 4 | 7 | 3 | 2.30 | 1.70 | 1.39 | 0.00 | 16.71 | 40 |
| data5-4-4 | 4 | 7 | 4 | 2.15 | 1.78 | 1.49 | 0.00 | 31.63 | 40 |
| data5-4-5 | 4 | 7 | 5 | 2.43 | 1.77 | 1.51 | 0.00 | 20.08 | 40 |
| data3-5-3 | 5 | 5 | 3 | 1.15 | 0.82 | 0.66 | 0.00 | 19.99 | 40 |
| data3-5-4 | 5 | 5 | 4 | 1.03 | 0.96 | 0.89 | 0.00 | 10.50 | 40 |
| data3-5-5 | 5 | 5 | 5 | 1.00 | 0.87 | 0.78 | 0.00 | 12.71 | 40 |
| data4-5-3 | 5 | 6 | 3 | 2.18 | 1.09 | 0.69 | 0.00 | 31.91 | 40 |
| data4-5-4 | 5 | 6 | 4 | 1.85 | 1.47 | 1.25 | 0.00 | 23.89 | 40 |
| data4-5-5 | 5 | 6 | 5 | 1.55 | 1.46 | 1.24 | 0.02 | 25.53 | 40 |
| data5-5-3 | 5 | 7 | 3 | 2.60 | 1.60 | 1.25 | 0.00 | 24.91 | 40 |
| data5-5-4 | 5 | 7 | 4 | 2.53 | 1.71 | 1.45 | 0.00 | 17.53 | 40 |
| data5-5-5 | 5 | 7 | 5 | 2.58 | 1.93 | 1.58 | 5.30 | 22.48 | 40 |
| data3-6-3 | 6 | 5 | 3 | 1.50 | 1.04 | 0.85 | 0.00 | 15.35 | 40 |
| data3-6-4 | 6 | 5 | 4 | 1.28 | 1.00 | 0.91 | 0.00 | 15.89 | 40 |
| data3-6-5 | 6 | 5 | 5 | 0.55 | 0.91 | 0.87 | 0.00 | 3.92 | 40 |
| data4-6-3 | 6 | 6 | 3 | 2.98 | 1.51 | 1.05 | 0.00 | 51.46 | 39 |
| data4-6-4 | 6 | 6 | 4 | 2.35 | 1.28 | 0.99 | 0.20 | 34.62 | 39 |
| data4-6-5 | 6 | 6 | 5 | 2.28 | 1.85 | 1.57 | 0.46 | 28.36 | 39 |
| data5-6-3 | 6 | 7 | 3 | 4.05 | 1.72 | 1.17 | 3.59 | 46.60 | 39 |
| data5-6-4 | 6 | 7 | 4 | 2.78 | 1.97 | 1.60 | 1.17 | 38.73 | 39 |
| data5-6-5 | 6 | 7 | 5 | 2.28 | 2.17 | 1.86 | 3.75 | 27.45 | 40 |
| data6-6-3 | 6 | 8 | 3 | 3.83 | 2.53 | 2.00 | 1.90 | 35.88 | 39 |
| data6-6-4 | 6 | 8 | 4 | 3.48 | 2.70 | 2.27 | 4.13 | 34.38 | 40 |
| data6-6-5 | 6 | 8 | 5 | 2.85 | 2.98 | 2.55 | 2.62 | 25.16 | 40 |
| data10-6-3 | 6 | 12 | 3 | 8.05 | 5.35 | 4.42 | 1.95 | 36.61 | 31 |
| data10-6-4 | 6 | 12 | 4 | 6.38 | 5.37 | 4.72 | 0.62 | 46.48 | 32 |
| data10-6-5 | 6 | 12 | 5 | 4.68 | 5.11 | 4.53 | 0.23 | 20.93 | 34 |
| data3-7-3 | 7 | 5 | 3 | 1.98 | 1.10 | 0.87 | 0.02 | 22.58 | 40 |
| data3-7-4 | 7 | 5 | 4 | 2.05 | 1.45 | 1.18 | 0.00 | 29.31 | 40 |
| data3-7-5 | 7 | 5 | 5 | 1.65 | 1.28 | 1.13 | 0.00 | 13.81 | 40 |
| data4-7-3 | 7 | 6 | 3 | 3.18 | 1.60 | 1.21 | 0.00 | 40.85 | 40 |
| data4-7-4 | 7 | 6 | 4 | 3.03 | 1.87 | 1.55 | 0.03 | 35.24 | 39 |
| data4-7-5 | 7 | 6 | 5 | 2.33 | 2.05 | 1.72 | 0.41 | 31.65 | 40 |
| data5-7-3 | 7 | 7 | 3 | 3.08 | 1.75 | 1.60 | 0.08 | 12.96 | 38 |
| data5-7-4 | 7 | 7 | 4 | 3.65 | 2.49 | 2.05 | 0.07 | 42.91 | 35 |
| data5-7-5 | 7 | 7 | 5 | 2.80 | 2.44 | 2.08 | 1.61 | 25.87 | 40 |
| data3-8-3 | 8 | 5 | 3 | 1.88 | 0.93 | 0.83 | 0.00 | 9.04 | 40 |
| data3-8-4 | 8 | 5 | 4 | 1.85 | 1.25 | 1.08 | 0.00 | 14.94 | 40 |
| data3-8-5 | 8 | 5 | 5 | 1.48 | 1.31 | 1.21 | 0.00 | 11.46 | 40 |
| data5-8-3 | 8 | 7 | 3 | 4.70 | 2.32 | 1.68 | 0.73 | 41.88 | 39 |
| data5-8-4 | 8 | 7 | 4 | 3.53 | 2.76 | 2.25 | 3.41 | 37.77 | 40 |
| data5-8-5 | 8 | 7 | 5 | 2.98 | 2.67 | 2.25 | 2.28 | 26.97 | 39 |
| data5-9-3 | 9 | 7 | 3 | 4.78 | 2.27 | 1.83 | 0.22 | 37.01 | 35 |
| data5-9-4 | 9 | 7 | 4 | 3.70 | 3.40 | 2.82 | 1.27 | 28.92 | 39 |
| data5-9-5 | 9 | 7 | 5 | 3.95 | 3.41 | 2.77 | 0.37 | 31.02 | 38 |
| data5-10-3 | 10 | 7 | 3 | 6.23 | 2.84 | 2.21 | 15.03 | 62.35 | 33 |
| data5-10-4 | 10 | 7 | 4 | 5.20 | 3.07 | 2.55 | 2.69 | 31.68 | 36 |
| data5-10-5 | 10 | 7 | 5 | 4.53 | 3.00 | 2.63 | 0.49 | 20.38 | 36 |
| data6-10-3 | 10 | 8 | 3 | 6.40 | 4.42 | 3.36 | 0.14 | 43.97 | 34 |
| data6-10-4 | 10 | 8 | 4 | 5.98 | 4.45 | 3.66 | 0.39 | 30.15 | 32 |
| data6-10-5 | 10 | 8 | 5 | 4.93 | 4.12 | 3.46 | 0.48 | 23.47 | 36 |
| data10-10-3 | 10 | 12 | 3 | 12.08 | 6.89 | 5.56 | 15.35 | 51.42 | 21 |
| data10-10-4 | 10 | 12 | 4 | 11.98 | 9.85 | 8.43 | 2.49 | 21.06 | 16 |
| data10-10-5 | 10 | 12 | 5 | 9.85 | 7.93 | 6.93 | 4.02 | 21.45 | 22 |
| Average: | | | | 3.22 | 2.32 | 1.94 | 1.23 | 26.23 | |

Furcy D, Koenig S (2005) Limited discrepancy beam search. *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 125–131, IJCAI'05.

Garey MR, Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (New York, NY, USA: W. H. Freeman & Co.).

**Table 11** Beam Search results for unequal retrieval probabilities.

| Beam Seach - Rule 01 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Instance Set | OccRate | E[Rel] | Time(s) | BGap(%) | Solved | Optimal | Less 1s |
| BRTP | 70% | 1.18 | 0.01 | 0.02 | 1079 | 1075 | 1078 |
| | 75% | 1.31 | 0.04 | 0.39 | 1079 | 1061 | 1075 |
| | 80% | 1.49 | 0.01 | 0.25 | 1080 | 1070 | 1078 |
| Caserta et al. (2012) | $60\% - 85\%$ | 1.90 | 1.25 | 1.13 | 2460 | 2377 | 2414 |
| Beam Seach - Rule 02 | | | | | | | |
| Instance Set | OccRate | E[Rel] | Time(s) | BGap(%) | Solved | Optimal | Less 1s |
| BRTP | 70% | 1.18 | 0.00 | 0.07 | 1080 | 1076 | 1080 |
| | 75% | 1.31 | 0.01 | 0.48 | 1080 | 1065 | 1079 |
| | 80% | 1.49 | 0.00 | 0.37 | 1078 | 1073 | 1078 |
| Caserta et al. (2012) | $60\% - 85\%$ | 1.85 | 0.37 | 1.91 | 2510 | 2379 | 2493 |
| Beam Seach - Rules 01 and 02 | | | | | | | |
| Instance Set | OccRate | E[Rel] | Time(s) | BGap(%) | Solved | Optimal | Less 1s |
| BRTP | 70% | 1.18 | 0.00 | 0.02 | 1080 | 1075 | 1080 |
| | 75% | 1.31 | 0.00 | 0.43 | 1080 | 1061 | 1080 |
| | 80% | 1.49 | 0.00 | 0.25 | 1080 | 1070 | 1080 |
| Caserta et al. (2012) | $60\% - 85\%$ | 1.85 | 0.01 | 2.00 | 2517 | 2377 | 2513 |

Gharehgozli AH, Laporte G, Yu Y, de Koster R (2015) Scheduling twin yard cranes in a container block. *Transportation Science* 49(3):686–705.

Kim KH (1997) Evaluation of the number of rehandles in container yards. *Computers & Industrial Engineering* 32(4):701–711.

Kim KH, Hong GP (2006) A heuristic rule for relocating blocks. *Computers & Operations Research* 33(4):940–954.

Kim KH, Kim HB (1999) Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics* 59(1-3):415–423.

Kim KH, Park KT (2003) A note on a dynamic space-allocation method for outbound containers. *European Journal of Operational Research* 148(1):92–101.

Lehnfeld J, Knust S (2014) Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research* 239(2):297–312.

Roberti R, Pacino D (2016) A decomposition method for finding optimal container stowage plans. *Operations Research* Submitted for publication.

Steenken D, Voß S, Stahlbock R (2004) Container terminal operation and operations research - a classification and literature review. *OR Spectrum* 26(1):3–49.

Vis IFA, Roodbergen KJ (2009) Scheduling of container storage and retrieval. *Operations Research* 57(2):456–467.

Watanabe I (1991) Characteristics and analysis method of efficiencies of container terminal - an approach to the optimal loading/unloading method. *Container Age* 36–47.

Zehendner E, Caserta M, Feillet D, Schwarze S, Voß S (2015) An improved mathematical formulation for the blocks relocation problem. *European Journal of Operational Research* 245(2):415–422.

Zhou R, Hansen EA (2005) Beam-stack search: Integrating backtracking with beam search. *In International Conference on Automated Planning and Scheduling (ICAPS)*, 90–98.